AFRL-IF-RS-TR-2004-288
Final Technical Report
October 2004

MORPHABLE COMPUTER ARCHITECTURES FOR HIGHLY ENERGY AWARE SYSTEMS

University of Notre Dame

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J869

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-288 has been reviewed and is approved for publication

APPROVED: /s/
WILMAR SIFRE
Project Engineer

FOR THE DIRECTOR: /s/
JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Oct 04	Final May 00 – Sep 03

4. TITLE AND SUBTITLE
MORPHABLE COMPUTER ARCHITECTURES FOR HIGHLY ENERGY AWARE SYSTEMS

5. FUNDING NUMBERS
C - F30602-00-2-0525
PE - 62301E
PR - MORP
TA - H0
WU - 01

6. AUTHOR(S)

Peter Kogge, Vincent Freeh, Jay Brockman, Kanao Ghose, and Nikzad Toomarian

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Notre Dame
Dept of Computer Science & Engineering
384 Fitzpatrick Hall
Notre Dame, IN 46556

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 N. Fairfax Drive
Arlington, VA 22203-1714

AFRL/IFTC
26 Electronic Pky
Rome, NY 13441-4514

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2004-288

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Wilmar Sifre, IFTC, 315-330-2075, Wilmar.Sifre@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

To achieve a revolutionary reduction in overall power consumption, computing systems must be constructed out of both inherently low-power structures and power-aware or energy-aware hardware and software subsystems. Today's most prevalent practices involve simple frequency scaling and modes where subsystems are merely powered on or off as needed. The energy expended per computational event is not as adjustable, even when lower than peak performance is acceptable. This is true as we move towards memory intensive hierarchical systems (such as register files, caches, SRAM, DRAM, Flash memory) where placement of data within the hierarchy has as much effect on energy expenditures as lowering the logic power. As modern processing systems begin to incorporate bigger and more complex storage hierarchies, it becomes imperative to incorporate techniques for managing such storage hierarchies in a manner that reduces the energy dissipation in the system as a whole. Power Aware architectures will provide a wide dynamic range in adjustable performance/energy settings, run-time software to dynamically manage these settings against real-time constraints, compilation techniques, programmer hints and run-time systems to control these settings or "gears". In essence, we need a system that morphs to meet the performance needs of the systems with the least amount of energy.

14. SUBJECT TERMS
Morphable Architecture, Power Aware, Energy Aware, Adaptive Algorithms, Voltage & Frequency Scaling, Dynamic Resource Allocation

15. NUMBER OF PAGES
45

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

List of Tables

1. Project Goals and Tasks

To achieve a revolutionary reduction in overall power consumption, computing systems must be constructed out of *both* inherently low-power structures *and* power-aware (or even better **energy-aware**) hardware and software subsystems. Today's most prevalent practices involve simple frequency scaling and modes where subsystems are merely powered on or off as needed. The energy expended per computational event (memory access or issuance of new instructions) is not, however, as adjustable, even when lower than peak performance is acceptable. This is particularly true as we move towards memory intensive hierarchical systems (such as register files, caches, SRAM, DRAM, Flash memory) where placement of data within the hierarchy has as much effect on energy expenditures as lowering the logic power. As modern processing systems begin to incorporate increasingly bigger and more complex storage hierarchies, it becomes absolutely imperative to incorporate techniques for managing such storage hierarchy in a manner that reduces the energy dissipation in the system as a whole, including both the processing ("logic") and memory components. To go significantly beyond the current state of the art requires architectures with a wide dynamic range in adjustable performance/energy settings, and run-time software to dynamically manage these settings against real-time constraints. Additional energy savings will also come from the use of compilation techniques, programmer hints and run-time systems to control these settings or "gears". In essence, we need a system that morphs to meet the performance needs of the systems with the least amount of energy.

Consequently, the Morph project had two major goals. The first goal was to develop morphable architectures whose energy/performance characteristics can be shaped to meet available energy profiles [KFG+ 00]. The second goal of the Morph project was to develop tools and software which will configure the system, estimate power/energy requirements of the system accurately and place and manage data within a system so that these energy saving techniques can be achieved.

The application area targeted by the Morph project was the general area of space applications for deep-space probes. Such applications have a large dynamic range of processing requirements: the processing and storage needs fluctuate dynamically as a function of the environment and the mission phase. Additionally, the energy sources for running these applications are highly constrained in a variety of ways, making it imperative to conserve as much power as possible for any and all computations that are performed.

A third and implicit goal of the Morph project was to constrain our solutions to the design space of superscalar datapaths. Superscalar processors offer binary compatibility and high performance. Other solutions to achieving a high-performance, such as the use of a multi-processing platform requires a thorough redesign of existing applications, run-time libraries and the operating system. Another reason for constraining our solutions to the space of superscalar CPUs had to do with the fact that the current and near-term space computing systems at JPL were based on a superscalar processor, the IBM PowerPC 750 (PPC 750). We expected to use a PPC 750-based testbed that was developed at JPL in the

course of our Phase I effort to provide data that could be used to validate our architectural and system-level solutions in depth as part of our Planned Phase II effort (which was not funded).

A fourth and another implicit goal of the Morph project was to concentrate on solutions that were technology-independent, as these are more universal in nature. It is also generally agreed up on that such higher-level, technology independent solutions have the potential for providing the highest degree of energy and power savings. All of the solutions developed by the Morph project are orthogonal to the traditional techniques that employ voltage and power scaling. Consequently, voltage and frequency scaling can be deployed in conjunction with our proposed scheme for reducing the energy and/or power requirements further.

The major tasks carried out in the course of the Morph project were as follows:

1. **Scenario Development**: Develop some target profiles of energy usage versus performance needs as a function of time based on some potentially real applications, and augment this with suggested benchmark programs.

2. **Morphable Architecture**: Develop a complete node organization and architecture whose performance/energy characteristics can be dynamically adjusted using a variety of energy-performance "gears".

3. **Baseline Characterization**: Characterize energy versus performance numbers for a baseline family of subsystems.

4. **Energy Aware Data Placement**: Develop some energy-aware data placement policies for mixed memory hierarchies.

5. **Adaptive Configuration Algorithms**: Develop algorithms which, given both expected performance/energy profiles and/or current run-time energy-relevant measurements, provide appropriate settings for the morphable architecture.

6. **Augmented Run-time**: Define and prototype as appropriate APIs for run-time systems for the morphable architecture which allows specification of data placement hints or explicit data movements within the hierarchy from the programmer or compiler, along with incorporation of real-time measurements to control system settings.

7. **Demonstration and Evaluation**: Using the simulator from Task 2, extrapolation of the energy/power gains of this architecture for a variety of potential design points.

As detailed in our quarterly reports and past annual reports, all of these tasks have been completed successfully. The bulk of the results developed as part of the Morph project have been published in 5 journal papers and more than 20 fully-reviewed and leading conferences in relevant areas. Two patent applications on inventions related to the Morph project have also been filed.

In this final report, our goal is to provide a concise summary of the major results of the Morph project and develop a set of recommendations for the design of future energy-efficient computing systems for deep-space probes. Many of the solutions developed in the Morph project are also applicable to mainstream superscalar processors and systems. Table I summarizes the main results pertaining to each of the seven tasks described above and also provides the "roadmap" information for this report.

Task number/ description	**Relevant solutions/activities**	**Where described in this report**
1. Scenario development	a) Energy/performance measurements for actual space applications from PPC750-based testbed developed for this project b) Energy-performance measurements for baseline system from the "Accupower" simulator developed in this project	Sections 2 and 3
2. Morphable Architecture	a) Zero-byte encoding for processor, storage and interconnections b) Energy/power reduction of the issue queue c) Dynamic allocation of datapath resources d) Exploitation of short-lived variables in reducing reorder buffer and register file complexity/power e) Variable-cluster superscalar datapath f) Inherently low-power circuit components	Sections 4.1 through 4.7
3. Baseline characterizations for a number of configurations	Integrated into evaluation of techniques associated with Task 2.	Sections 4.1 through 4.7.
4. Energy-aware data placement	a) Compile-time data placement. b) Dynamic data movement within register-caches and ROB integrating physical registers	Sections 5, 4.5 and 4.6.
5. Adaptive configuration algorithm	a) Dynamic allocation of datapath resources b) Linux run-time system implementation for matching energy-performance characteristics	Sections 4.2 and 6.
6. APIs for morphing/adaptation/placement	Integrated into solutions developed in Tasks 4 and 5.	Sections 4, 5 and 6
7. Demonstration and evaluations	Integrated into the evaluations of the techniques developed in Tasks 2, 4 and 5	Sections 2, 3, 4, 5 and 6

Table 1. Tasks, Results and Roadmap for the Report

To provide a more logical framework for this final report, we break down and present the main results of the Morph project into the following parts:

- The PPC750-based Testbed and Measurements on the Characteristics of Space Applications
- The Accupower Toolset for Energy and Performance Estimations
- Microarchitectural-Level Morphing Techniques for Energy-Efficiency
- Energy-Aware Data Placement
- Run-time System to Tune Energy/Performance Characteristics
- Final Results and Recommendations

A bibliography listing the main publications resulting from this report is included at the end of this report.

2. A Testbed for Characterizing Applications for Deep-Space Probes

As part of the Morph project, we have developed a testbed of a future deep space mission, DEEP IMPACT, and instrumented it to provide insight into very realistic processing loads and timing requirements under a variety of scenarios. Our original project goal was to use this data to baseline time and power data against which our proposed new architectures would be, and have been, evaluated. However, the wealth of data from this testbed is useful for far more than just our project, since they represent a unique view into a type of application that will become of increasing importance, namely embedded systems involving multiple different tasks running under real-time constraints.

2.1 Testbed Features

The energy-estimation testbed itself is modified single board computer (SBC) of the same type and performance as is projected to fly in several future missions [KNA+03a, KNA+ 03b]. Modifications to the board include chip extenders to allow access to individual chips. This testbench was then connected to a testbed that provided a simulation of the rest of the spacecraft, in real time. A methodology was then developed that allowed instrumentation connected to this testbench to perform significant and detailed power and timing measurements, while application scenarios were being run.

We opted for a single-board computer (SBC) designed by WindRiver Systems that housed a Motorola PowerPC 750 processor, 128 Mbytes of RAM, a serial port, and two PCI ports. It runs VxWorks^{tm} as its operating system, just as the real system. One PCI port contained a Network Interface Card, while the other PCI port contained a reflective shared memory card. This board was a prime choice due to its use of a PGA (Pin Grid Array) connection to attach the processor chip to the board. This allowed us to access all the pins on the processor, and thus directly measure current between the power and

ground pins on the processor. Furthermore, access to all the I/O pins allows the use of a logic analyzer to accurately obtain a timing profile of the processor's interaction with peripheral devices, e.g. bridge controllers, memory, etc.

Modifications were made to the SBC to allow access to circuit connections designated for power to the processor. An oscilloscope with a current probe measured the current being drawn by the CPU and memory. A power distribution module provided easy access via a connector to the board housing the processor and memory. Calculating power from the current measurements was simply done by measuring the voltage at the CPU and memory and using the product of the two to determine the instantaneous power consumption. We refer to this testbed as the “power measurement” testbed.

2.2 Monitoring Facilities on the Testbed

JPL supports an Autonomous Test Bench, known as Babybed, which had been previously been developed for mission software development and performance benchmarking. The system runs on two processors that share a common VME or PCI back plane. One processor runs a simulation of a virtual vehicle and the second processor runs the control software that drives the simulated vehicle, currently a three-axis stabilized, free-floating spacecraft. The simulation is a fully end to end real-time software simulation which allows actual mission fight software to communicate with virtual spacecraft devices like actuators and sensors, as well as power, telecom and science hard-ware. As commands are issued and executed by the virtual devices, the spacecraft dynamics are affected accordingly and the sensor models sense the results.

The autonomous test bench also includes two single-board computers communicating via shared memory implemented using two reflective memory cards and a fiber optical cable providing transparent communication between them. One of the SBCs is responsible for running the flight software. The other single-board computer is responsible for running a space environment simulator as inputs to the flight software. A performance profile was generated for the flight software in terms of cache misses, CPU utilization, and MIPS.

The test bench described in Section 2.1 was ported to the autonomous test bench by replacing the single-board computer running the flight software with the power measurements test bench. Because the power measurements test bench provided two PCI slots, a reflective memory card with a PCI interface was used to enable communication of the space environment simulator with the power measurement test bench.

2.3 Power Measurement Methodologies

Power profiling was performed at three levels of granularity. At the fine-grain level, power was measured at the instruction-level. The instruction set of the processor used for experiments, a Motorola PowerPC 750, was profiled comprehensively, i.e. a measurement for each instruction in the instruction-set was obtained. The basic idea for profiling was to place a single instruction into a loop, and repeat it until it reaches a

steady state. At this point we could measure the power of that instruction.

At the medium-grain level, power was measured at the event-level. Events at the micro-architecture level included data-forwarding, cache accesses, floating-point pipelining, instruction units, etc. A sequence of instructions was programmed with the understanding that certain events would take place. For example, sequencing an add instruction with dependencies on each previous add instruction would invoke the data-forwarding mechanism to avoid pipeline stalls.

Lastly, coarse-grain measurements were performed at the operating system level by analyzing multiple running tasks and associated power profiles with the tasks. This paper focuses on these results.

To identify tasks that were executing during a test we invoked a WindRiver tool called WindView. This allowed us to generate a timeline of software activity. The start and stop time of each task when it took control of the processor was shown graphically and could also be stored in a database file. Combined with the power measurements, the relationship between the power profile of software and the tasks running were trivial.

Power profiles were taken from a subset of the entire software suite. The modules we profiled included image compression, orbit determination, maneuver planning, and several operating modes. The first three mentioned are of importance because they show dynamic profiles that reflect their computational complexity.

The timing profiles produced by WindView became very useful in determining which task was consuming power at different times. There is a close correlation between power profiles and timing profiles.

2.4 The Application Suite

The application suite used in Morph is derived from the Deep Space 1 (DS1) mission, and modified to reflect potential use on the Deep Impact mission [KNA+ 03b]. Deep Impact involves a comet orbiter that includes an instrumented spike-like probe. After release from the orbiter, the probe will impact the comet. Visual analysis of the dispersal from the orbiter should yield significant insight into the interior nature of the comet.

The DS1 flight software is comprised of some 60 tasks that are initiated and initialized at startup, and run forever. The tasks vary in priority. Some tasks wake up and execute in response to interrupts, and others run when the scheduler activates them. The software operates loosely on a 1 second cycle. Nominally all tasks run at least once every second. Some tasks run more frequently (4Hz and 8Hz rates). The general procedure is for a task to wake up, run and go back to sleep (pending state).

When a task wakes up, it performs some basic duties and/or check its inter-process communications (IPC) queue(s). It will attempt to complete its duties and/or process all the data in its queue(s) before it is suspended by the operating system. Any unfinished

business is picked up the next time the task awakes. Naturally, tasks with high priority have more opportunity to execute each second than lower priority tasks. As a result, busy high priority tasks may prevent lower priority tasks from ever waking up within some second.

When there is not much activity and all tasks finish early within the 1 second interval, the tIdleTask uses the remaining time. As system activity increases, the amount of time the tIdleTask runs becomes less and less.

Due to ITAR restrictions, the DS1 flight software is not publicly available to the research community. As an alternative, a software module from the suite was modified so that it could be made publicly available. The module chosen, MICAS, is an image compression algorithm. The computational complexity coupled with the need for generous amounts of memory I/O made this a good choice for power profiling and optimization.

2.5 Simulated Flight Scenarios and Measurements

Seven different flight scenarios were profiled for power consumption and time traces. The first scenario (prelaunch) simulated the flight computer while in an idle state prior to vehicle launch. The second scenario was to profile the flight after a vehicle launch. The third and fourth scenarios (DSEU_Scan and DSEU_Burst) involved placing the flight computer into different modes corresponding to their DSEU operation. The remaining scenarios involved image compression, orbit determination, and maneuver planning.

The Motorola PowerPC 750 can be configured to enable access to an L2 cache or to bypass the L2 and go directly to memory. In addition, a dynamic power management (DPM) feature, based on clock gating, can be enabled or disabled. Most of the seven scenarios mentioned above were run in three modes: L2 enabled with DPM on, L2 enabled with DPM off, and L2 accesses disabled with DPM off.

The traces as gathered from the testbench formed the basis for the load analysis of the Deep Impact code done for the Morph project. This analysis was done in three steps - a preprocessing step, an interval identification and analysis step, and a power modeling step.

The trace logs consisted of literally thousands of entries, each giving start time and task number. These were first inspected to eliminate recording errors, and then preprocessed to remove effects of the task tracing hooks by replacing all routines known to be part of the testbench system, and not to be found in the expected flight software, by idle. Multiple neighboring entries to the same task (primarily idle) were then collapsed into one, and each remaining trace entry augmented from outside information with the task priority (in the current state of the flight software, priority assignments are static, and range from a highest level of “0”, to a low idle level of “255”). A variety of global statistics were then computed for each trace, such as total time, tasks started per second, total busy and idle time, etc., along with statistics on each task, such as the number of

times it was run, the minimum, maximum, and average execution times, and the standard deviation in this time.

From each preprocessed trace, sequences of processing "intervals" were identified, where one interval starts at the end of an idle period, after which a string of one or more real tasks were executed, and terminated with some other idle period. For each such interval, the overall busy and idle periods were computed. Histograms were then developed from this data.

A key statistic taken from each such interval for use in the power modeling analysis was the ratio of the total interval time to the processing time. This ratio indicates by how much a factor the CPU's performance could be slowed down during the processing period to eliminate the idle period, and run in a lower power mode.

2.6 Application Characteristics

The testbed was used to investigate the characteristics of each of the 15 scenarios. The results are also summarized in [KNA+ 03b]. Some of the parameters measured were:

1) the total time of the simulated mission phase,

2) the total number of original traces from the trace log, expressed as an average per second,

3) the equivalent number of scheduling "events" per second after eliminating the non-operational tasks and combining idle tasks,

4) the average number of intervals per second that would be observed in the mission phases,

5) the percent of the total scenario time that the microprocessor was idle,

6) the percent of the time that the processor was busy executing code (100% minus the prior number).

Figure 1 represents an example of the measurements made using the testbed and shows three of the more important of these characteristics grouped by mission phase, and

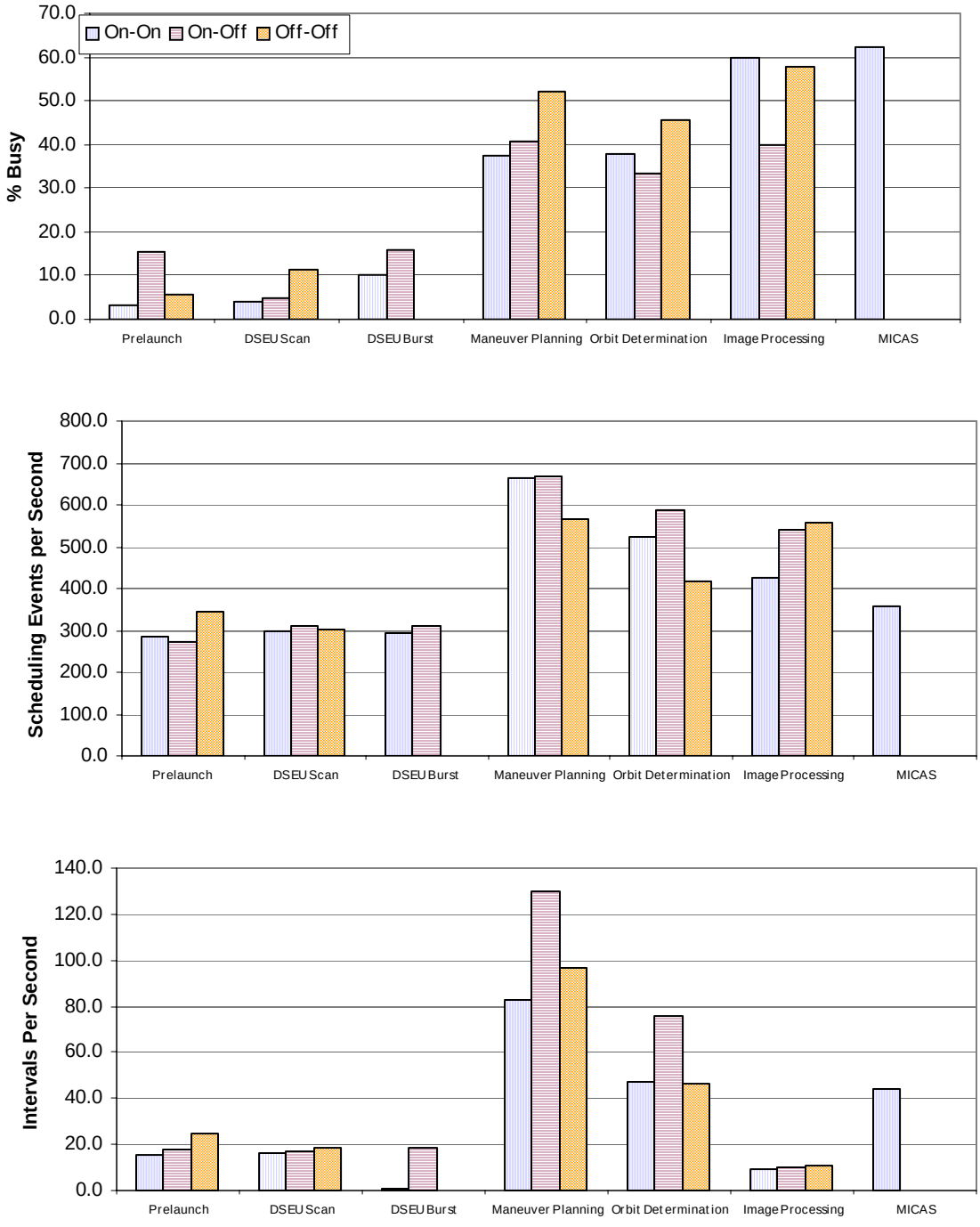


Figure 1. Sample measurements from the Deep Impact Suite

ordered in rough order of load on the CPU (% Busy). In all of the measurements depicted in Figure 1, three measurements were made for each mission phase: both L1 and L2 caches on ("on-on"), L1 cache on and L2 cache off ("on-off") and both caches turned off ("off-off"). Some of the main conclusions from our measurements are as follows:

- A key observation is the broad range of busy times, from about 3% to 60%, for a 20 to 1 variation. This verified a key premise for the Morph program as a whole that processing loads were highly variable during missions, and that knowing that fact, and having control over power/performance of the CPU could lead to dramatic overall mission power and energy savings.

- One interesting observation that the data doesn't always play out as one would expect is variations as the settings for L2 and DPM are changed. For example, one would expect the most busy configuration to be the ones where L2 if off, but this is not true for Prelaunch. A similar seeming inconsistency can be seen in the "Intervals per second," where variations are strongest for the middle phases when the DPM is turned off.

- A more general observation is that for the most part, the number of tasks scheduled per second changes dramatically versus mission phase (as expected), but does not change much versus configuration. This makes sense, since in each phase one would expect the same mix of tasks to be executed at the same rates, regardless of the loading on the microprocessor.

To summarize, the analysis of the trace data for the Deep Impact application reveals the presence of a large dynamic range of the processing activities across the various phases of this application. CPU utilizations in-between idle periods can vary dramatically; the number of scheduling events within each phase as well as the number of intervals per second within a phase can both vary significantly from one phase to another. Understanding the dynamics of these statistics will be invaluable in understanding how best to craft efficient real-time embedded processing systems, especially for space applications. Implications of the characteristics of the Deep Impact applications in designing an energy-efficient space computing systems are described in [KNA+ 03].

3. The AccuPower Toolset

The AccuPower toolset [PKG 02] was developed as part of this project to accurately measure the energy-performance characteristics of baseline configurations and to evaluate the efficacy of the various techniques developed in this effort for enhancing the energy-efficiency of superscalar processors.

Several power estimation tools for processors have been designed, including Wattch [BTM 00], Simplepower and TEM^{2}P^{2}EST [CAI 99] to name a few. Simplepower is used only for simple 5-stage scalar pipelines and only models the execution of integer instructions; it is not applicable to superscalar processors. The major drawback of other tools is their reliance on the Simplescalar simulator [BA 97], which lumps many critical

datapath artifacts like the issue queue (IQ), the reorder buffer (ROB) and physical register files (PRFs) into a unified structure called Register Update Unit (RUU), quite unlike the real implementations, where the number of entries and the number of ports in all these structures are quite disparate. Consequently, power dissipation cannot be estimated accurately on a per-component basis. Considering that in many cases, these components collectively contribute to more than half of the overall power dissipation of the chip, it is necessary to have facilities that allow the design space of these components and their interactions be modeled as accurately as possible. This is exactly one of the areas where AccuPower provides a more accurate simulation/power estimation framework than existing tools. Specifically, the main features of the AccuPower toolset are as follows:

Detailed cycle-level simulation of all major datapath components and interconnections that mimic the actual hardware implementation, including separate and realistic implementations of the issue queue, register files, reorder buffers, load-store queues and forwarding mechanisms.

- Detailed and accurate simulations of the on-chip cache hierarchy (including multiple levels of on-chip caches, interconnections, arbitration and chip-level I/O traffic).

- Built-in models for three major variants of superscalar processor designs that are in wide use.

- Well-instrumented facilities for collecting datapath statistics of relevance to both power and performance at the level of bits, bytes (for data and instruction flows) within logic blocks and subsystem-level components and the entire processor.

- Implementations of cutting-edge techniques for power/ energy reduction at the microarchitectural level, logic level and circuit level, as well as techniques based on clock gating, voltage and frequency scaling to facilitate the exploration of the design space.

- Use of energy/power dissipation coefficients for energy dissipating events within datapath components derived from SPICE measurements of actual layouts of these components. These coefficients are used in conjunction with transition counts obtained from the microarchitectural simulation component of AccuPower to accurately estimate the power/energy dissipations. Coefficients for leakage dissipations are also provided.

We believe that short of an actual implementation, AccuPower's power estimation strategy is as accurate as it gets.

3.1 Datapath Styles Supported

Three widely used design variations of superscalar processors that are somewhat prevalent are modeled in the AccuPower toolset. These datapath variations are as follows:

Datapath A: Here, input registers that contain valid data are read out while the instruction is moved into the IQ. As the register values required as an input by instructions waiting in the IQ (and in the dispatch stage) are produced, they are forwarded through forwarding buses that run across the length of the IQ. The dispatch buffer entry for an instruction has one data field for each input operand, as well as an associated tag field that holds the address of the register whose value is required to fill the data field. When a function unit completes, it puts out the result produced along with the address of the destination register for this result on a forwarding bus. Comparators associated with each IQ entry then match the tag values stored in the fields (for waited-on register values) against the destination register address floated on the forwarding bus. On a tag match, the result floated on the bus is latched into the associated input field. Since multiple function units complete in a cycle, multiple forwarding buses are used; each input operand field within an IQ entry thus uses a comparator for each forwarding bus. Examples of processors using this datapath style are the IBM Power PC 604, 620 and the HAL SPARC 64.

Datapath B: Here, even if input registers for an instruction contain valid data, these registers are not read out at the time of dispatch. Instead, when all the input operands of an instruction waiting in the IQ are valid and a function unit of the required type is available, all of the input operands are read out from the register file (or as they are yet to be written to the register file, using bypassing logic to forward data from latter pipeline stages) and the instruction is issued. In this case, the IQ entry for an instruction is considerably narrower compared to the IQ entries for Datapath A, since entries do not have to hold input register values. The dispatch/issue logic can be implemented using a global scoreboard that keeps track of instructions and register/FU availability. Alternatively, an associative logic similar to that of Datapath A can be used to update the status of input registers for instructions waiting within the IQ. Examples of processors using this datapath style are the MIPS 10000, 12000, the IBM Power 3, the HP PA 8000, 8500, and the DEC 21264.

Datapath C: Here, the ROB entry set up for an instruction at the time of dispatch contains a field to hold the result produced by the instruction - this serves as the analog of a physical register. We assume that each ROB entry may hold only 32-bit long result, thus requiring the allocation of two ROB entries for an instruction producing a double-precision value. A dispatched instruction attempts to read operand values either from the Architectural Register File (ARF) directly (if the operand value was committed) or associatively from the ROB (from the most recently established entry for an architectural register, in case the operand value was generated but not committed. Source registers that contain valid data are read out into the DB entry for the instruction. If a source operand is not available at the time of dispatch in the ARF or the ROB, the address of the physical

register (i.e., ROB slot) is saved in the tag field associated with the source register in the IQ entry for the instruction. Forwarding to the waiting IQ slots is performed similar to Datapath A. Examples of processors using this datapath style are the Intel Pentium II and Pentium III.

3.2 Implementation Details

The AccuPower tool consists of the three components: a microarchitectural simulator, which is a greatly modified version of the Simplescalar; the VLSI layouts for major datapath components and caches in a 0.5micron process; and power coefficients obtained from the SPICE simulations.

To support the three superscalar datapath configurations discussed in section 3.1, we significantly modified the Simplescalar simulator [BA 97] and implemented separate structures for the DB, the ROB, the rename table, the physical register files and the architectural register files. Three versions of the simulator were designed - one for each datapath configuration discussed above. We also accurately modeled the pipeline structures and various interconnections within the datapath, namely the dispatch buses, issue buses, result buses and commit buses. In a typical superscalar processor, multiple sets of such buses are needed to sustain the dispatch/issue/commit rate. Traffic on each such bus and read/write activity within the register files implementing the datapath storage components are separately monitored and analyzed as discussed later.

The extent of our modifications to the Simplescalar can be gauged by the fact that barely 10% of the original code was retained. The sim-outorder.c file was completely rewritten to support a true cycle-by-cycle out-of-order instruction execution. This is in contrast to the original Simplescalar code, where instructions are actually executed in-order at the dispatch stage and the effects of out-of-order execution are achieved through the convoluted manipulations with the RUU. Significant modifications have also been incorporated into the cache simulator, as discussed below. Specifically, the changes that were made to the Simplescalar to simulate the realistic superscalar pipelines are as follows:

(i) We split the monolithic cache access stage as used in Simplescalar into two stages to mimic the real-world situation where cache accesses - even L1 cache accesses - are typically performed in multiple cycles and provided a support for pipelined cache.

(ii) We modeled the contention for the bus between L1 and L2 caches. This is important because the L1 caches are typically split with separate caches for instructions and data. L2 cache, however, is typically unified. The situation when both I1 miss and D1 miss occurs in the same cycle is certainly possible and they require proper modeling to arbitrate for the access to the L2 cache.

(iii) Along similar lines, we modeled the contention for the off-chip interconnection from the L2 cache to the DRAM modules.

(iv) The decode stage of the Simplescalar datapath, where instruction dispatching and register renaming is performed, was split into two stages, as it is unrealistic to perform the fairly complicated operations of dispatching, register renaming and source register readout in a single cycle. The only exception to this is Datapath C, due to the use of associative addressing for performing a source physical register lookup instead of an indirect addressing through a rename table.

(v) We also assumed realistic delays on the interconnections, noting that it takes a full cycle to distribute the result produced by one of the FUs to the waiting operands in the IQ.

(vi) For Datapaths A and B, the branch misprediction handling was implemented by waiting until all instructions prior to the mispredicted branch commit, after which the contents of the ROB, the IQ, the physical register files and the rename table were flushed and the execution was restarted from the correct PC. Of course, the branch misprediction penalty for Datapaths A and B was higher than in Simplescalar implementation. For Datapath C, we tagged each instruction with the id of the preceding branch and selectively flushed the ROB and the IQ on a misprediction by using the associative search for the appropriate tags. This is doable for the Datapath C, because the ROB incorporates rename table and physical register files. For Datapaths A and B, such selective flushing results in the inconsistent values of the valid bits stored within physical register files.

To summarize, we attempted to design a simulator that would closely mimic the actual microarchitecture and hardware implementations of real CPUs on a cycle-by-cycle basis. The focus of the AccuPower tool is to facilitate the exploration of the design space of superscalar processors and gauge the impact of well-used and cutting-edge techniques for saving power and/or energy. The tool also supports the exploration of circuit-level techniques and the more standard power reduction techniques like voltage and frequency scaling.

3.3 VLSI Layouts

For the estimating the energy/power for the key datapath components using AccuPower, the transition counts and event sequences gleaned from the microarchitectural simulator were used, along with the energy dissipations for each type of event, as measured from the actual VLSI layouts using SPICE. CMOS layouts for the on-chip caches, DB, PRF, ARF and DB in a 0.18 micron 4 metal layer CMOS process were used for the key datapath components to get an accurate idea of the energy dissipations for each type of transition.

The register files that implement the ROB and IQ were carefully laid out to optimize the dimensions and allow the use of a clock speed of up to 1 GHz. A Vdd of 3.3 volts is assumed for all the measurements. (The cache determined the value of the clock cycle time, as obtained from the cache layouts for a two-stage pipelined cache in the same technology.) In particular, these register files feature differential sensing, limited bit line driving and pulsed word line driving to save power. Augmentations to the register file

structures for the IQ (mainly in the form of comparators with each of the 3 source operand fields and the four result/tag buses) were also fully implemented; a pulldown comparator was used for associative data forwarding to entries within the IQ and the device sizes were carefully optimized to strike a balance between the response speed and the energy dissipations. For each energy-dissipating event, SPICE measurements were used to determine the energy dissipated. These measurements are used in conjunction with the transitions counted by the hardware-level, cycle-by-cycle simulator to estimate dynamic energy/power accurately. Actual layout data was also used for estimating the leakage power of the layouts in the smaller feature sizes.

3.4 Speeding up the AccuPower Tool - Multithreading

The instrumentation needed to determine the bit level activities within data flow paths and data storages (both explicit and implicit) and log all major switching activities slows down the simulation drastically. To get reasonable overall simulation performance with all the instrumentation in place, we resorted to the use of multithreading. Specifically, we use a separate thread for the data stream analysis. The two-threaded implementation is run on SMPs to get an acceptable level of simulation speed - one that approached close to that of the original Simplescalar without the heavy instrumentation. The data acquired from basic instrumentation within the main simulation thread is buffered and fed into a separate thread where it was analyzed for the lack of entropy within significant byte slices and all byte slices within a data item as well across consecutive data items within a data stream.

With a single thread implementing all of the simulation, instrumentation and analysis, the overall simulation speed was reduced by as much as 40% compared to the original Simplescalar simulation without any modification and instrumentation. With both threads in place, and with the use of inter-thread buffers of an optimized size, the overall simulation time achieved was often significantly better on a SMP compared to the single-threaded implementation. The performance of the dual-threaded version was also acceptably close to that of the original Simplescalar simulator without any of the enhancements and the instrumentation.

The AccuPower toolset was extensively used in the bulk of our studies related to tradeoffs between power and performance in the relevant baseline configurations as well as to evaluate the improvements in the overall energy-efficiency of the processor when the morphing techniques described in the next section were used.

4. Microarchitectural-Level Morphing Techniques for Energy-Efficiency

As part of the Morph project, we investigated microarchitectural-level morphing techniques for dramatically enhancing the overall energy-efficiency of a superscalar processor in the following three dimensions:

- At the level of bit-slices and byte-slices: the technique developed for this dimension exploited the common occurrences of zeros in the higher-order bits of results and stored data [GPK+ 00].

- At the level of datapath components: the techniques developed here improved the energy-efficiency of large datapath structures/subsystems like the issue queue [KGPK 01, PKE+ 03c], the load-store queues, caches, register files and reorder buffers [KEP+ 03a, PKE+ 04b, KPE+ 04a]. One of the techniques developed in this category dynamically activated and deactivated regions of some key datapath components to adapt the datapath to the dynamic characteristics of the applications [PKG 01].

- At the level of clusters: the main technique developed in this category was an adaptive technique for a multi-clustered datapath where the issue rate and the number of clusters were dynamically adapted to match the characteristics of the applications [ZK 01].

In virtually all of these solutions, relevant circuit-level requirements, where needed, were also evaluated in depth. We now describe these solutions in some depth.

4.1 Squeezing Out Zeros: Selective Byte-Slice Activation

The datapath width of a typical superscalar processor (currently at 32 bits or 64 bits) is grossly underutilized in many situations. This is because many data values - often as much as 50% - flowing on the datapath or stored in the various implicit or explicit storage structures like register files, caches, issue queue, reorder buffer have leading zeros that do not contribute to precision/significance of the data. Such a behavior stems from the use of small literal values, byte operations implemented with masking and shifting, use of extended precision formats, use of zero padding within unused fields of instructions etc. By encoding bytes containing all zeros using a single additional bit per byte, significant energy savings can be realized within function units [BM 99], within instruction caches [VZA 00], and within superscalar datapaths [GPK+ 00]. The circuit techniques used for implementing such encoding and decoding of bytes containing all zeros are simple, and result in energy savings within all datapath components and interconnections [GPK 01]. It is also shown in [GPK+ 00] and [GPK 01] why it makes sense to encode a run of zeros at the level of bytes instead of encoding a run of arbitrary number of consecutive zeros in the data.

Figure 2 shows several representative plots justifying the feasibility of zero byte encoding for a typical superscalar datapath. In Figure 2 (a), the percentage of leading zero bytes and zero bytes throughout the operand, weighted and averaged over 32-bit and 64-bit operands, are shown for individual SPEC 95 benchmarks. For the integer benchmarks, which are dominated by mostly 32-bit integer data streams, roughly 47% of the leading bytes are all zeros, while about 49% of the bytes contain all zeros (including all zero bytes in the leading bit positions). For the floating point benchmarks, which consist of 32-bit integers, 32-bit floats and 64-bit doubles, the weighted average shows that roughly

32% of the leading bytes are all zeros while 37% of the bytes in the data streams contain all zeros. The floating point numbers account for the higher difference between the number of leading zero bytes and the number of zero bytes throughout the operands. Figure 2 (b) shows the distribution of zero bytes throughout the operands for some specific data flow paths (the dispatch, issue, result and the commit buses.) Considerable energy savings will be achieved within all datapath components such as caches, register files, the dispatch buffer, the re-order buffer and function units if the presence of leading bytes with all zeros and bytes containing zeros throughout the operands are exploited.

In Figure 2 (c), we show what happens when the zero valued bytes are not driven on transfer paths and, in addition, bit values within the non-zero valued bytes that do not change from their prior-driven values on the interconnection (driven from the same source or a different sources) are also not driven. The combined amount of bit slices that do not have to be driven in this case is in the range of 65% to 81% on the aforesaid buses. This result suggests that a considerable amount of power can be saved in transferring data on these interconnections, particularly as wire capacitances begin to dominate in implementations that have smaller feature sizes.

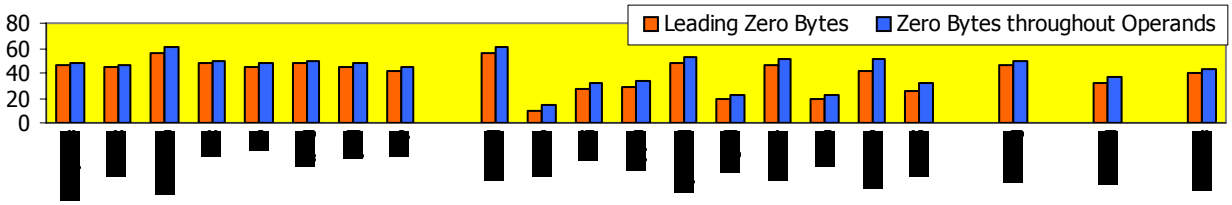


Figure 2a. Percentage of zero bytes averaged over all operand sizes and all data streams

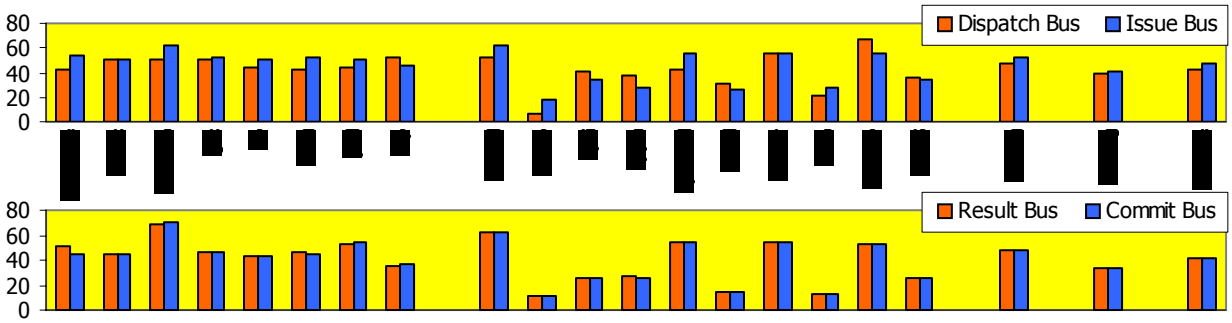


Figure 2b. Percentage of zero bytes throughout operands on internal buses

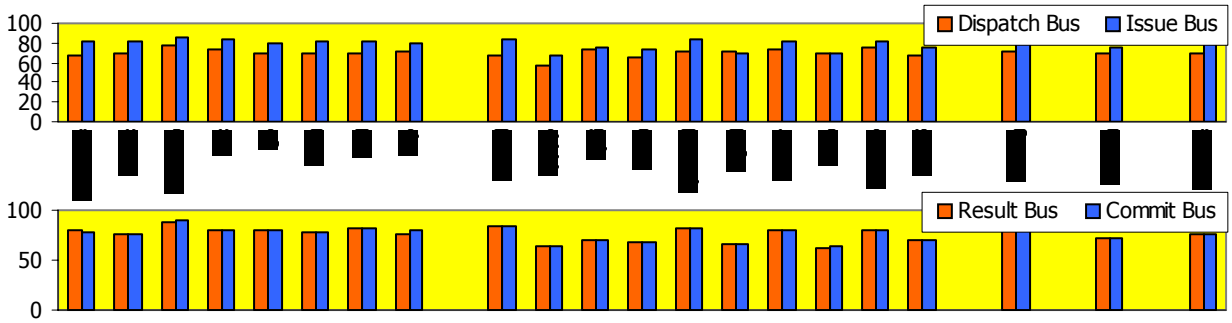


Figure 2c. Percentage of bit slices not driven when zero bytes are encoded and bit positions that did not change in value from the last cycle are not driven

4.2 Adapting Datapath Resources to Match Program Dynamics

The “one-size-fits-all" philosophy used in allocating datapath resources results in the overcommitment of datapath resources like register files, the issue queue and the reorder buffer. Such structures have power dissipations that grow as polynomial functions of size, number of ports, and need for associative search. Our approach to minimizing the power requirements of the datapath is to use a dynamic resource allocation strategy that tries to closely track the actual dynamic resource demand of the executing program [PKG 01]. In a typical superscalar datapath, controlling the allocation of a single datapath resource such as just the issue queue - is not sufficient for realizing a high degree of energy efficiency. This is because of the correlation among the usage of resources such as the issue queue (IQ), reorder buffer (ROB - integrating the physical register file,) and the load/store queue (LSQ), where load and store instructions are dispatched.

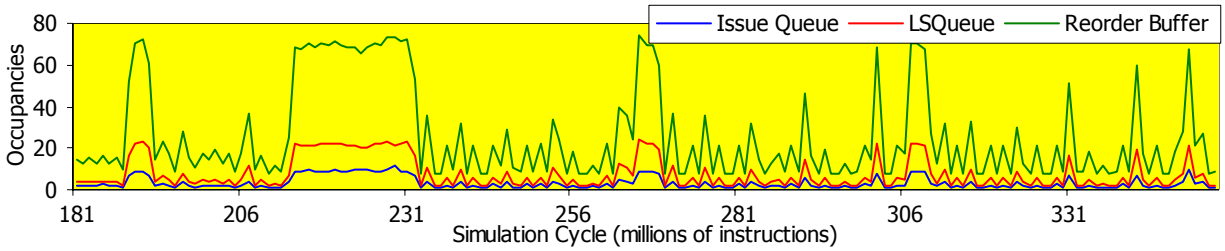


Figure 3a. Occupancies of the IQ, the ROB and the LSQ (fpppp)

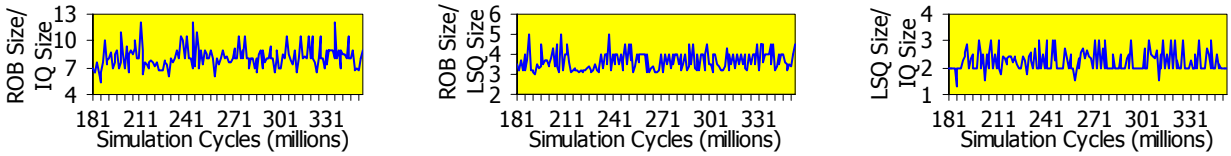


Figure 3b. Datapath resource occupancy ratios (fpppp)

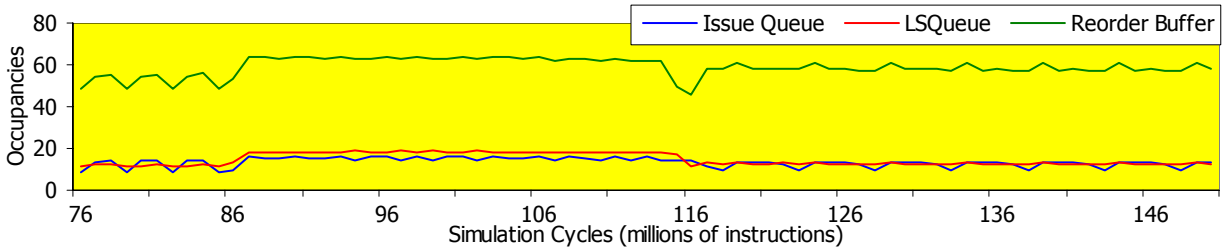


Figure 4a. Occupancies of the IQ, the ROB and the LSQ (ijpeg)

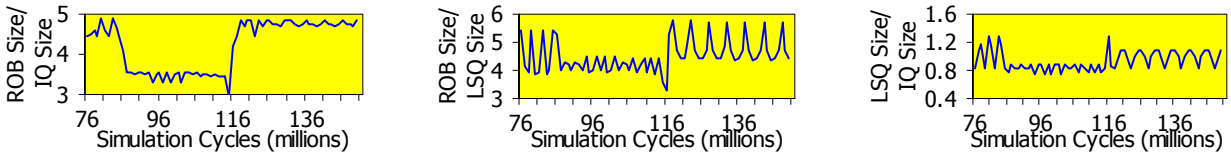


Figure 4b. Datapath resource occupancy ratios (ijpeg)

We studied the correlations among the occupancies (number of valid entries) of the IQ, the ROB and the LSQ in the context of our experimental framework. Representative results for one integer (*ijpeg*) and one floating point (*fpppp*) benchmark from SPEC 95 suite are shown in Figures 3 and 4. Figures 3 (a) and 4 (a) show the occupancies of the three resources for *fpppp* and *ijpeg* benchmarks respectively, and Figures 3(b) and 4(b)

show the ratios of these occupancies. Measurements were taken for 200M committed instructions after skipping the first 200M. For each benchmark, we recorded and cleaned the average occupancies after every 1 million committed instructions and then graphed the results.

As seen from these graphs, the occupancies of the three resources are positively correlated. This suggests that resizing only one datapath resource is insufficient - in fact, if the sizes of other resources are not dynamically adjusted as well, these resources will remain overcommited most of the time. Another observation from Figures 3 and 4 is that it is difficult, if not impossible, to adjust the sizes of multiple resources by monitoring one of these resources and rescaling the number of active partitions within other resources proportionately. This is primarily because the ratios of the resource occupancies also change drastically across program's execution. For example, for *fpppp* benchmark, the ratio of the ROB occupancy to the IQ occupancy varies between 4 and 12 in the time interval shown (Figure 3 (b)). One can consider resizing schemes, where the actual resource occupancies and their ratios are periodically sampled for a short duration and the appropriate prescaling coefficients are then set. However, this would result in the complication of control logic and besides, monitoring hardware would still be needed for every structure anyway, even though it would only be used during the set-up periods. For these reasons, we opted to perform the independent monitoring of individual datapath resource usages and use this information to dynamically adjust the size of each resource separately. In the next section, we explain the details of our resizing strategy.

We primarily investigated the use of simple techniques to resize the IQ, the ROB (integrating physical registers) and the LSQ independently. In particular, these components are partitioned and the number of active (i.e., powered up) partitions is chosen dynamically to closely track the actual demands of the program. Downsizing the components by shutting off one or more active partitions is done at periodic intervals, at the end of an update period. The actual occupancies (i.e., number of allocated units) is measured several times within an update cycle and its average at the end of an update period is used to drive the downsizing decision: if the average occupancy is lower than the total combined capacity of the currently active partitions by one or more partitions worth, then the overcommitted partitions are marked for turning off, with the actual turnoff taking place later when allocated entries within these marked partitions are used up/consumed. By sampling the resource usage periodically (instead of on a continuous basis, as done in [BAS+01, FG 01]) for the issue queue only, we conserve dissipations within the monitoring logic for the downsizing. Our results also reveal that the error arising from the computations of the average occupancy by taking the average occupancy measured at discrete points (at every sample interval) is tolerable since significant energy savings are achieved using our approach. Although the results given here shows the savings on dynamic/switching power, dynamic deallocation of partitions also saves leakage power that would be otherwise dissipated within the IQ, ROB and LSQ.
We also compensate for any inaccuracy in estimating the average occupancy of a resource by taking a very aggressive approach for resizing up a resource. An additional partition is made available as soon as the number of overflows with the currently allocated partitions crosses a threshold value - we do not wait until the end of the

sampling or update interval to perform this upsizing. Our results show that as a consequence of doing this aggressive upsizing, the dynamic resource allocation results in very low performance degradation. The overflow counters used to drive such upsizing are activated only on overflows and their small width (a few bits) allow energy dissipations within the upsizing/monitoring logic to be kept very small. Dynamic resource allocation, as described, relies on the use of traditional implementations of the IQ, ROB and the LSQ with simple enhancements for supporting incremental resource allocations, as described in [PKG 01].

Figure 5 (from [PKG 01]) documents the power savings possible through the use of dynamic resource allocation schemes for a 4-way (= peak issue rate of 4 instructions per cycle) and 6-way superscalar processor. The energy savings realized within the IQ, ROB and the LSQ are quite significant - about 40% to 60% on the average in the IQ, 52% to 65% on the average in the ROB and 38% to 40% on the average in the LSQ. (The savings shown factor in power expended within the control and sensing logic for resizing.) Further power savings, up to an additional 15% can be realized on top of the savings due to resizing if zero byte encoding and circuit innovations are employed [GPGK 01]. These savings are realized with a minimal impact on the layout area and with a performance penalty of less than 4% on the average.

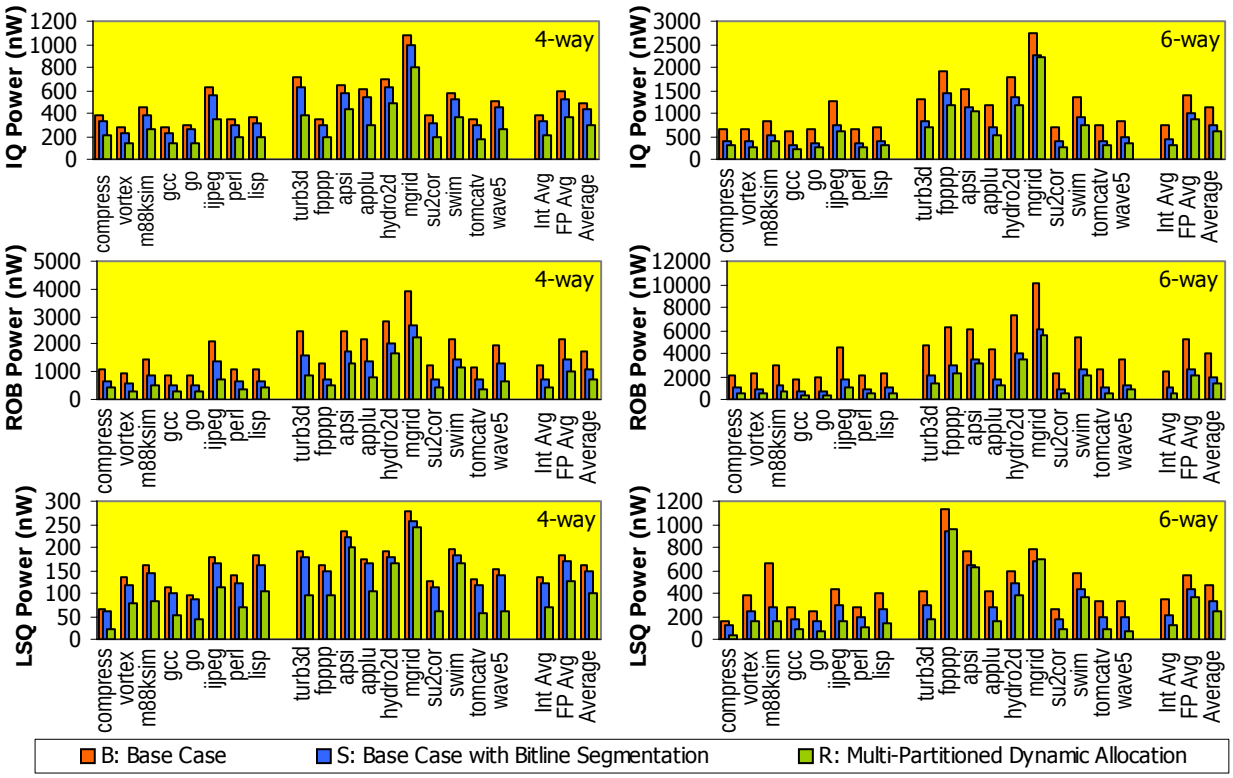


Figure 5. Power savings in the IQ, ROB and LSQ for 4-way and 6-way processors with adaptive resizing

4.3 Low-Power Cache Design

The bulk of the energy/power dissipation within a cache memory system occurs in the course of reading and writing bitcells. Our main approach for reducing the dynamic power dissipation within a cache relies on the use of multiple line buffers, as introduced in [KG 99] to avoid the readout of data from the cache tag/data arrays.

In caches with multiple line buffers, shown in Figure 6, substantial power savings result when the line accessed can be obtained from one of the line buffers, allowing the access of the tag and data arrays to be aborted, thereby saving power. In these caches, the tag and data array accesses are started simultaneously with the comparison of the set number of the currently accessed address with the set numbers of the lines stored in the line buffers. The cache timing with the line buffering mechanism is shown on the right half of Figure 6. This is done to avoid any potential extension of the cache cycle time. The normal set of tag comparators is used to detect a hit or miss irrespective of where the cache line comes from - from a line buffer or from the tag/data arrays. In effect, the line buffers act as a smaller cache that is accessed in parallel with the cache. In addition to the line buffers, modest investments are made for latches to hold the set numbers associated with the contents of each line buffer and comparators to compare the set numbers stored in these latches with the set number for the data being accessed. The line buffered caches are further enhanced by the use of subbanking and bit line segmentation as described in [Itoh 96].

How does zero byte encoding help the on-chip caches? To get an answer to this, we studied conventional on-chip cache organizations as well as on-chip caches with multiple line buffers, subbanking and bit-line segmentation as described in [KG 99]. We assumed the use of 2 and 4 line buffers and bit line segments spanning 16 consecutive rows for all of the on-chip caches. The results of adding zero byte encoding on top of these artifacts, averaged over all the SPEC 95 benchmarks, is shown in Figure 7. Notice that the L1 D-cache dissipates more energy than the L1 I-cache; this is because of the higher sense amp dissipations that are inevitable in the 4-way set associative L1 D-cache (as opposed to a sense amp dissipation of one-fourth the amount in the L1 I-cache; for the same associativity, the L1 I-cache dissipates more energy than the L1 D-cache in general). We

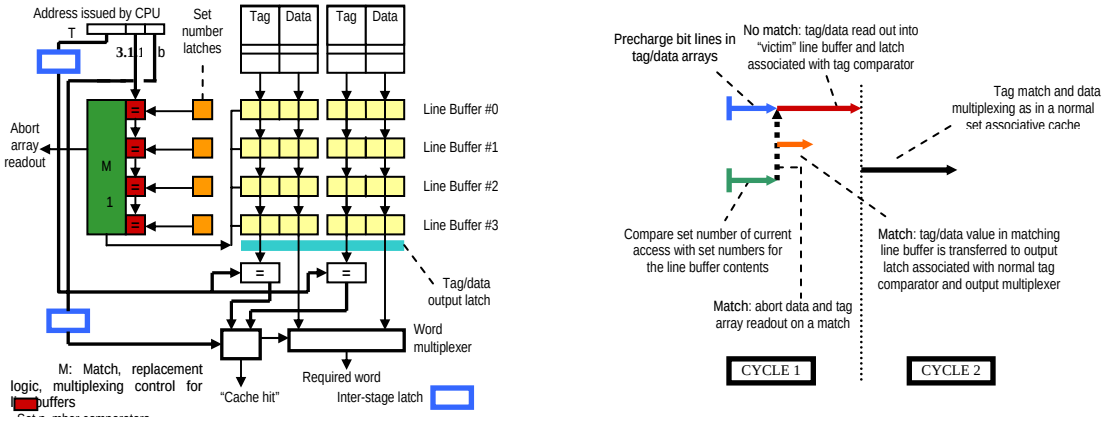


Figure 6. A 2-way set-associative cache with 4 line buffers & its timing

assume that as cache lines are filled from memory, zero bytes are dynamically encoded. The L2 cache is assumed to be on-chip.

As Figure 7 shows, the L1 I-cache benefits very little from *leading* zero byte encoding (about 5% or so), with the benefits coming from the encoding of the unused literal field of register-to-register instructions. The L1 D-cache benefits more substantially from zero byte encoding, more particularly for conventional designs than for a design using 4 line buffers. The L2 caches also benefit from zero byte encoding. With 4 line buffers, bit line segmentation and zero byte encoding, the total dissipation in the cache hierarchy comes down to about 415 mW from about 2.2 Watts, representing an overall dynamic power savings of about 75%. The proposed Morph design uses 8 line buffers and zero encoding of all bytes within the caches to save additional power.

As part of the Morph project, we also investigated the use of morphing the cache structure and selective activation and deactivation of regions within the cache to conserve dynamic as well leakage power. Leakage dissipations are substantial at low feature sizes and with the use of lower supply voltages. Industry estimates often indicate that leakage dissipations in memory structures can almost equal the dynamic dissipations [BD 99]. Our main approach in reducing leakage dissipations in the caches relies on the selective activation and deactivation of regions within each cache. A region is essentially made up of a number of consecutive rows or sets. A region could be active (= powered up and dissipating leakage power, normal access time), or in the standby mode (powered up to retain contents, but with substrate bias applied to reduce leakage, slower access time) or completely powered down (using a high Vt device to gate the supply line). Accesses to cache regions are dynamically monitored to decide whether a region should be in one of the three modes. Our ongoing effort indicates that the approach described here results in a leakage power reduction in excess of 70% with less than 5% penalty on the performance. Another more obvious approach will be to apply a reverse substrate bias to the entire cache array - this is not an attractive solution, since cache cycle times are prolonged and a substantial amount of power is expended in maintaining the reverse substrate bias. As a bonus, selective activation of cache regions also reduce dynamic power through the use of bypass connections to allow bitlines to “skip" the standby or shut-off regions, thereby reducing bit line capacitance.

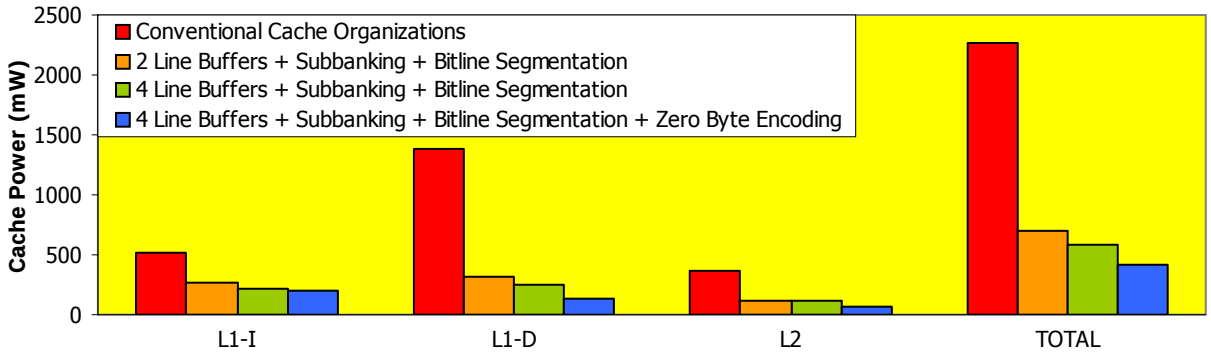


Figure 7. Dynamic power savings in the on-chip caches

A variation of these techniques can also come into play as chips move to the processing-in-memory time frame, and significant memory macros can be placed on the chip with the CPU core(s). Besides greatly reducing the energy per access (no off chip drivers and receivers, and better use of available bandwidth), the already present row latches within the memory macro can be used in fashions identical to that of the line buffers above. 10% or more of the already reduced energy of such macros can be saved. We also showed that the line-buffering technique described for caches can also be used to reduce the energy requirements of many register-file based structures in a superscalar processor, such as the rename table.

4.4 Issue Rate Modulation and Variable Cluster Microarchitecture

A key part of the Morph project was the development of microarchitectures for superscalar CPUs where the intrinsic performance of the core (in Instruction Per Clock (IPC)) can be varied dynamically, where the relationship between IPC and Energy Per Clock (EPC) is better than the 4-th power, and thus where the energy expended per cycle can be throttled back by large factors in times where performance is not important. The key to achieving this was to break what typically are common, hugely multi-ported, structures within a conventional superscalar CPU into multiple relatively independent clusters. Prior work on such "multi-cluster" superscalar microarchitectures [ZK01] demonstrated designs with the EPC at the same IPC as the best conventional design, or 20% more IPC at the same EPC. In the Morph project we carried this multi-cluster microarchitecture one step further to allow dynamic control over the number of clusters that are active. This allows a system to throttle back on performance needs during mission phases where performance is not important, and ride the intrinsic energy gains. For those mission portions needing only light processing, energy savings approaching an order of magnitude appear possible.

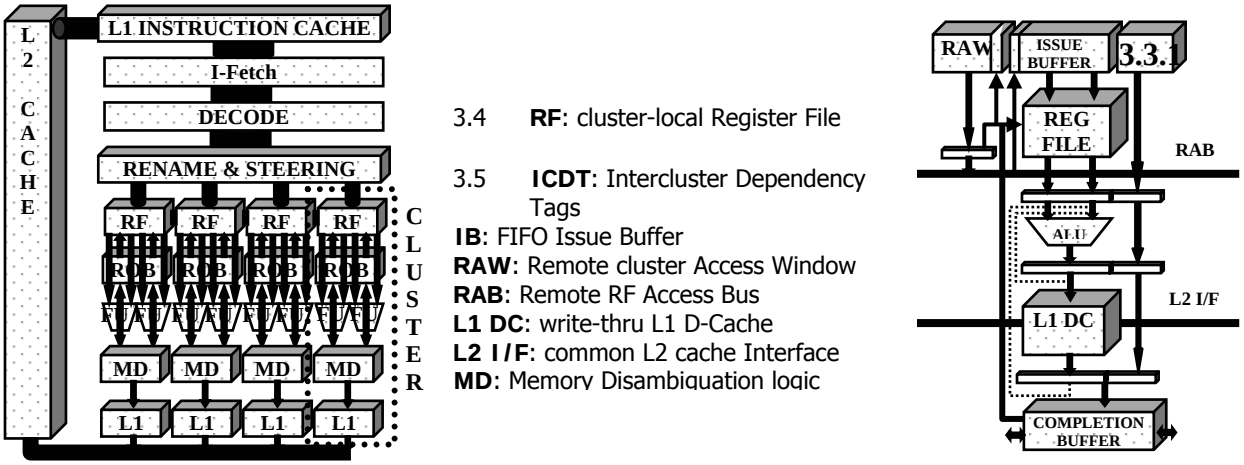


Figure 8. Multi-Clustered Morph Architecture

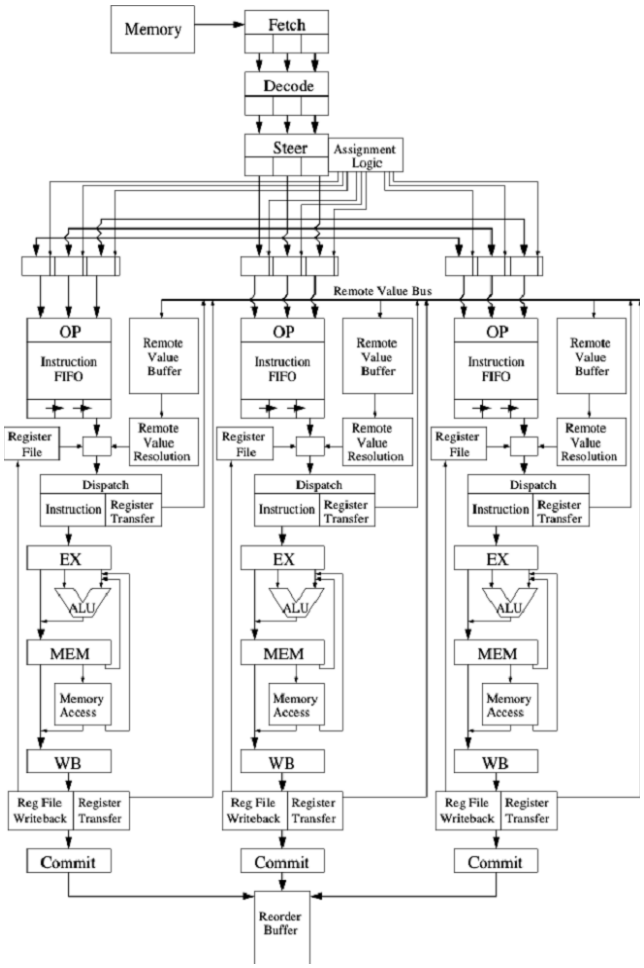
The multi-cluster microarchitecture, depicted in Figure 8, offers a substantial savings over a conventional superscalar microarchitecture - up to a factor of 2 in energy per

instruction (EPI) at the same performance (as measured by IPC). This, by itself, is substantial, and worth including. However, as with conventional designs, the resulting EPI is fixed, and significantly higher than that for a simple embedded-class core. The energy required to complete a program is thus fixed, even though changing the clock may change the power dissipated.

In the Morph effort, this multi-cluster technique was extended to allow the number of clusters to be varied dynamically, thus affecting both IPC and EPI. Simulations indicated that a 4 cluster design, which can be ratcheted down to 2 or 1 cluster, has an IPC that varies by a factor of 1.7 and an EPI that varies even more - by up to 3.4. Thus, in times of light performance needs, using the single cluster configuration may require up to 70% longer to compute a program at the same clock, but at roughly 1/3 the expended energy. If even lower IPCs are acceptable, then additional significant EPI savings are possible, including: shrinking the size of the caches, switching the cache timing from single cycle to two cycle pipelined, reducing or eliminating branch prediction, reducing or eliminating store buffers, or switching programs or data from normal RAM to lower energy/access but possibly higher latency RAMs. Given the appropriate circuit library, lowering the clock then allows Vdd to be scaled also, slowing the execution (but not changing the IPC) but reducing the EPI even further. Another 2X in EPI is thus possible.

In the original multi-cluster research, competing at the very highest performance levels was the goal, and thus each cluster was a low-level superscalar machine by itself, with significant complexity. This includes significant issue windows and physical register files, with significant register renaming (but still less than conventional). In Morph, however, we target applications where for large periods of time very little performance is needed. EPIs in the range of what can be achieved by the very best of "32-bit embedded controllers" are in order.

A revised Morph microarchitecture, termed RuDRA, was developed (Fig. 9) and a detailed behavioral simulator constructed. These simulations used instruction traces generated for a PowerPC processor. A simulated L2 cache, when present, was 512K 2-way set associative. L1, when present, was comprised of one 8-way set associative 4K cache bank per cluster. These matched numbers for the PPC testbed at JPL.



IPC	Clusters			
Clock	1.000	2.000	4.000	8.000
1.000	0.784	1.128	1.653	1.837
0.875	0.787	1.135	1.680	1.842
0.750	0.813	1.270	1.729	1.920
0.625	0.841	1.330	1.860	1.889
0.500	0.871	1.338	1.939	1.963
0.375	0.875	1.332	1.941	1.965
0.250	0.908	1.433	2.030	1.997
0.125	0.943	1.441	2.101	1.819

Figure 9. Revised RuDRA Multi-Cluster Architecture

The clusters themselves were simple in-order pipelined processors. This avoids most of the potentially costly comparisons of out of order execution, simplifies internal cluster design, and reduces power. A cluster contains a (simple) register file, ALU, and a slice of cache/memory access logic.

Because each cluster contains its own register file, there is no centralized register file. Instead, each cluster contains a subset of program-visible register values. Instructions which are dispatched to, and complete in, that cluster write back to the cluster-local register file. These cluster register files can be greatly simplified compared to a centralized register file, and require far fewer ports. Ideally, instructions dispatched to a cluster only require register values in the cluster's local register file. However, to maximize performance and load balancing, it may be necessary to use values in another cluster. The Remote Value Bus (RVB) allows such inter-cluster communication.

A “steering” stage dispatches instructions to the clusters using a simple heuristic which aims to maximize performance by partitioning instructions to minimize the amount of inter-cluster communication, and maximize the amount of parallel execution. This steering stage keeps track of which cluster has the most current value of a register, and

uses this data to partition instructions and to instruct clusters when they are required to make inter-cluster communication.

The first pipeline stage of the cluster is the operand fetch (OP) stage. This stage is responsible for gathering the register operands required for an instruction and resolving inter-cluster communication by transferring register values to and from the RVB. The steering stage instructs an OP to perform such transfers through an internally generated register transfer (RT) instruction. The OP stage sorts incoming instructions into a dispatch queue, and register transfer instructions into a separate queue. Both queues are serial (FIFO), to avoid out of order issue within the cluster. In any cycle, the OP stage may issue one RT or one normal instruction. A normal instruction may issue if their operands can be found in the local cluster register file, or if it has already been transferred across the RVB.

The Memory Access stage accesses the banked cache of the cluster. Each bank of this cache contains a segment of the address space. Cache accesses are assumed to be in-order, but logic at each bank detects any out of order memory collisions (RAW hazards) between loads and uncommitted stores. If such a collision is detected, the pipeline and any uncommitted stores are flushed.

For evaluation, the behavioral simulator was run against a series of benchmarks, including the MICAS program discussed earlier. It was run assuming 1, 2, 4, or 8 clusters, and a variety of L2 and main memory latencies to simulate accurately different clock rates. Each run gave an IPC, which when multiplied by clock rate gives a performance number. The 4 cluster, longest latency, numbers matched relatively well with the IPC from the PowerPC used in the testbed, and thus performance for the other configurations was ratioed to this value. An initial power model was then used to create expected dissipation for each configuration.

The table accompanying Fig. 9 gives the IPC numbers that resulted from the simulation, as a function of both relative clock rate and number of clusters. The lower clock rates reflect the reduced memory latencies, which in turn increased the IPC numbers, as discussed above. The data for the 8 cluster configuration doesn't behave exactly as the others do, primarily because of quirks in the inner loop of the code being simulated, and how it fit within the instruction blocks being fetched.

4.5 Reducing the Complexity of Reorder Buffers

As part of the Morph effort, we investigated a variety of techniques for reducing the overall complexity and power dissipation in the reorder buffer (ROB), one of the most power-consuming components of a superscalar datapath. The first solution investigated capitalized on the observation that the bulk of the data dependencies – in excess of 84% are satisfied in the course of data forwarding or in the course of reading committed results at the time of dispatching. Figure 10 shows the relevant percentages for a 4-way superscalar machine, as described in [KPG 02, KPE+ 04a]. This fact can be well

exploited in a datapath design that employs a separate architectural register file to hold committed values and where a reorder buffer that integrates the physical register file is

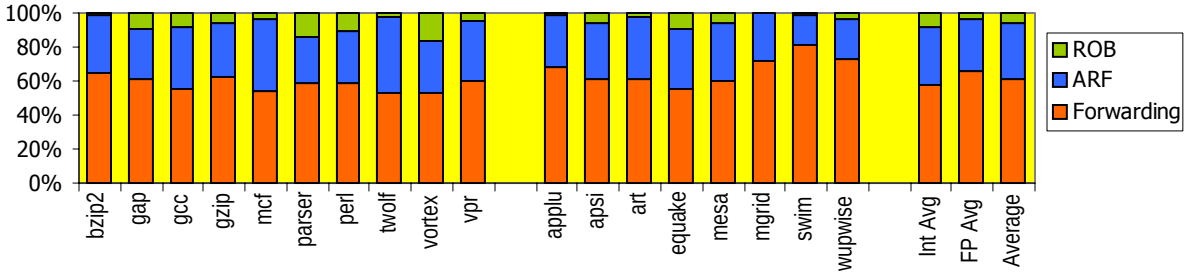


Figure 10. Breakdown of Acquisition of Source Operands

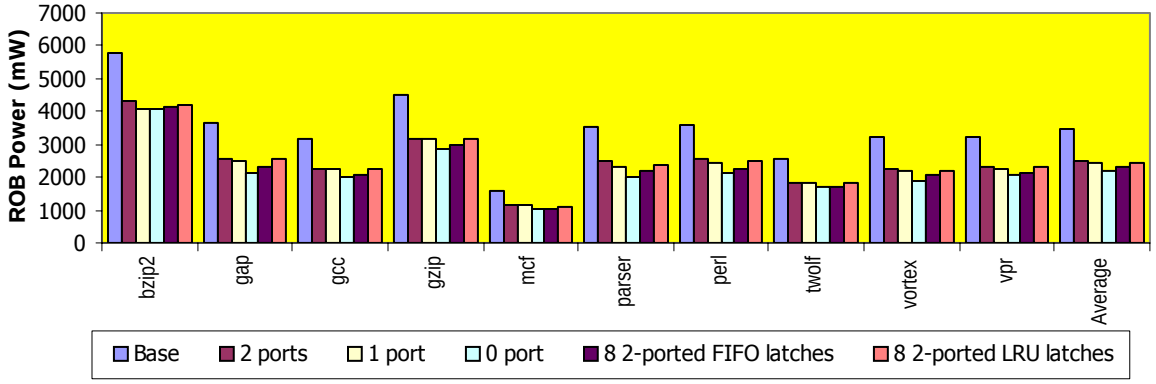


Figure 11. Power Savings from the Use of Recent-Value Caches

used to hold results that have not been committed. The main idea here is to cache a few recently generated results in a small register file (made up of a set of latches, using a LRU algorithm or a FIFO replacement algorithm) in an attempt to satisfy the demand of instructions that have to read a value that has not yet been committed. With this small register-cache in place, the ports for reading non-committed values can be eliminated completely from the ROB. For the few instructions that could not obtain a non-committed value from the ROB or in the course of forwarding, the existing set of forwarding buses are used to forward the result for a second time at the time of committing the result. With the use of a 8-entry result cache, a non-negligible power savings – as high as 20% - results with only a 4% performance degradation. Details of this technique are described in [KPG 02]. Figure 11 shows the power savings for a 72-entry ROB, as detailed in [KGP 02].

A second approach for minimizing the energy dissipation in the ROB was explored in [KPE+ 03b, KPE+ 04a]. Here, power savings result from a distribution of the ROB structures across the function units in a manner similar (but not identical) to what is used in multi-clustered datapaths and using register-caches with each distributed instance.

4.6 Exploiting Short-Lived Variables

A significant fraction of the results produced in a modern superscalar processor happen to be short-lived in the sense that by the time that the value is produced, the corresponding destination architectural register has been renamed. In excess of 80% of

the results are short-lived, for the processor configuration reported in [PKE+ 03b]. This fact can be exploited in reducing the energy dissipation of a superscalar datapath as follows. First, short-lived results – which are detected one cycle prior to their writeback, can be kept in a small cache and they do not need to be written to the larger ROB or physical register. Second, the values so cached do not need to be written to the architectural register file, thus saving additional energy. The only reason for holding on to the short-lived values in the small cache has to do with the handling of branch mispredictions or for implementing precise interrupts. As shown in [PKE+ 03b], the energy savings in the ROB/ARF can be about 20% or so for typical configurations.

4.7 Associated Circuit-Level Techniques to Improve Energy Savings

As part of our Phase I Morph effort, we have also devised new circuits that add to the energy efficiency of a superscalar datapath. One of these is a comparator that dissipates energy predominantly only on a match of the bit patterns being compared [KGPK 01, PKE+ 03c]. Traditional comparators (including those in content-addressable RAMs - CAMs), on the other hand, employ circuit designs that dissipate energy on a mismatch in any bit position. Comparators are used extensively in modern superscalar datapaths to implement associative data forwarding, within issue queues in register mapping, to detect memory dependencies in writeback queues, etc. In all of these structures, the number of mismatches far outweighs the number of matches. Significant energy savings are thus possible with the use of our dissipate-on-match comparators. For example, we have shown in [KGPK 01] that power dissipations in the issue queue can be reduced by about 15% to 20% on the average with the use of our comparators.

Additional circuit-level techniques were also explored in the Morph project for improving the overall efficiency of the microarchitectural techniques discussed in Sections 4.1 through 4.6. These included the design of circuits for encoding data on buses for reducing energy dissipation in the interconnections, circuits for reducing the energy dissipations within TLBs, register files and caches. These circuits are described in many of the papers that relate to the techniques presented in Sections 4.1 through 4.4 and Section 4.6.

5. Intelligent Data Placement Modules

Additional power savings at the system level can be realized with the development of an intelligent data placement module that places and moves data intelligently in the memory/cache hierarchy to minimize power/energy dissipations. Embedded systems usually have a variety of memory components within the memory hierarchy: DRAM of various sorts, ROM, EEPROM, etc. All of these offer differing latency/energy per access characteristics. To date, embedded systems have treated such mixed hierarchies only in terms of data retention, or for speed, but not for power. Given that some technologies such as spin tunneling cells, or on-chip memory macros, may have better energy per access characteristics by literally orders of magnitude over conventional off-chip memory, techniques for trying to utilize such mixed memories need to be developed.

In the Morph project, the MICAS/ERIC benchmark was used with a Shade based tool to analyze the address trace put out by a CPU core with and without a local cache, and targeting two separate "classes" of memory parts: a fast but high power memory and a slower but more energy efficient part. The storage was assumed to be divided into pages of 4KB each, with whole pages allocated to one type of memory. Total memory traffic, instructions only, and data only were all considered. In these initial studies, latency was not a concern - the emphasis was on the potential for peak energy savings. We assumed we could statically allocate the N most frequently referenced pages to the low energy memory, and the rest to high-energy memory. Then by assuming a ratio of energy per access between the two memory types, effective energy savings was computed as a function of N, with different energy ratios from 2:1 to 16:1 as parameters. The results were striking - for as little as 80-100 pages (<400KB) of "low energy" memory (of any ratio) at least one-half of the difference between the high and low energy memories could be saved. For a 16:1 memory system, this is just about the energy of the conventional memory.

The impact of this on embedded systems is two-fold: even small amounts of very energy efficient memory added to a system can result in large energy savings, especially when the system is ratcheted down from peak performance for quiet periods as is possible with Morph, and second, to achieve this requires hooks in the run-time that permits identification of these key pages, and under what conditions they should be remapped.

In Phase II we will consider both adding on-chip memory and building a run-time that can so adapt. We will also investigate the future potential for integrating novel low energy memories being developed elsewhere in the Power Aware Computer and Communication (PACC) program such as by Nonvolatile Memories.

The microarchitectural-level techniques of Sections 4.5 and 4.6 for reducing the energy dissipation can also be viewed as a dynamic scheme that places data intelligently within the memory hierarchy of the datapath. The use of caches with multiple line buffers, as described in Section 4.3, can also be viewed as a technique that moves data within a memory hierarchy in an intelligent manner in response to the program dynamics.

An original goal in the Morph project was to design APIs to assist in the placement and explicit movement of data in the storage hierarchy. This was not done, as existing primitives for page allocation and mapping, pinning pages and memory copying could be used to support the intelligent data placement techniques.

6. Kernel-Level Facility for Controlling Energy-Efficiency

In the course of the Morph effort, we investigated a feedback based control technique for adapting the processor performance (through voltage and frequency scaling) for stretching the processing time to a target value, given an energy source whose capacity decreases with use (such as a battery) [MFK 02]. The goal is basically to scale down the energy requirement (and performance) of the processor to allow the processing to continue processing for a pre-specified period.

Formally stated, if E is the initial energy capacity of the battery and the targeted processing time is T, then given that the energy left in the battery is E_f, then the goal of the approach is to select an operating point with an average power dissipation of P such that $P.T \leq (E - E_f)$. The closer P.T approaches $(E - E_f)$, the better is the performance level. In reality, a few things complicate the implementation of this simple policy. First, applications are dynamic in nature and their power requirements change over time, so maintaining a steady power consumption level of P is difficult. Second, kernel facilities monitor power consumptions and battery capacity at discrete intervals. Last, but not the least, real processors have discrete operating points as only a finite combination of voltage and frequency settings are available.

The approach taken in our strategy is to measure the energy expended and the battery capacity on a continuous basis and adjust the operating point (and power/performance) of the processor to correct for deviations from the ideal power level of P as given by the above equation. What results is thus a feedback based power management approach. The choice of a damping factor of a suitable value is critical in such a system for guaranteeing stability. Details of the proposed scheme are described in [MFK 02], which also describes a prototype implementation on the Linux kernel on an AMD processor (that supports both frequency and voltage scaling). As reported in [MFK 02], the prototype system is successful in meeting the battery life objective over a wide range of workload.

Extensions to the proposed scheme for handling battery recharging and for handling real-time tasks are also described in [MFK 02].

7. Final Results

The large spectrum of solutions for improving the energy-efficiency of computing systems in the Morph project precludes the assessment of the impact of combining all of these techniques in a single system. Given this, we could only estimate the effect of only a handful of the microarchitectural and circuit level solutions in reducing the power dissipation of a typical processor. We will do so in two steps.

7.1 Datapath Alone

Figure 12 shows the result of factoring in the datapath changes– without the use of clustering and traditional frequency and voltage scaling techniques. The 3X average power savings result in less than 5% performance degradation. Voltage scaling and frequency scaling can add further savings, but will result in a lower than peak performance. At the system level, we expect another 2X to 3X power savings from the use of the data mapping techniques and the use of an intelligent run-time system.

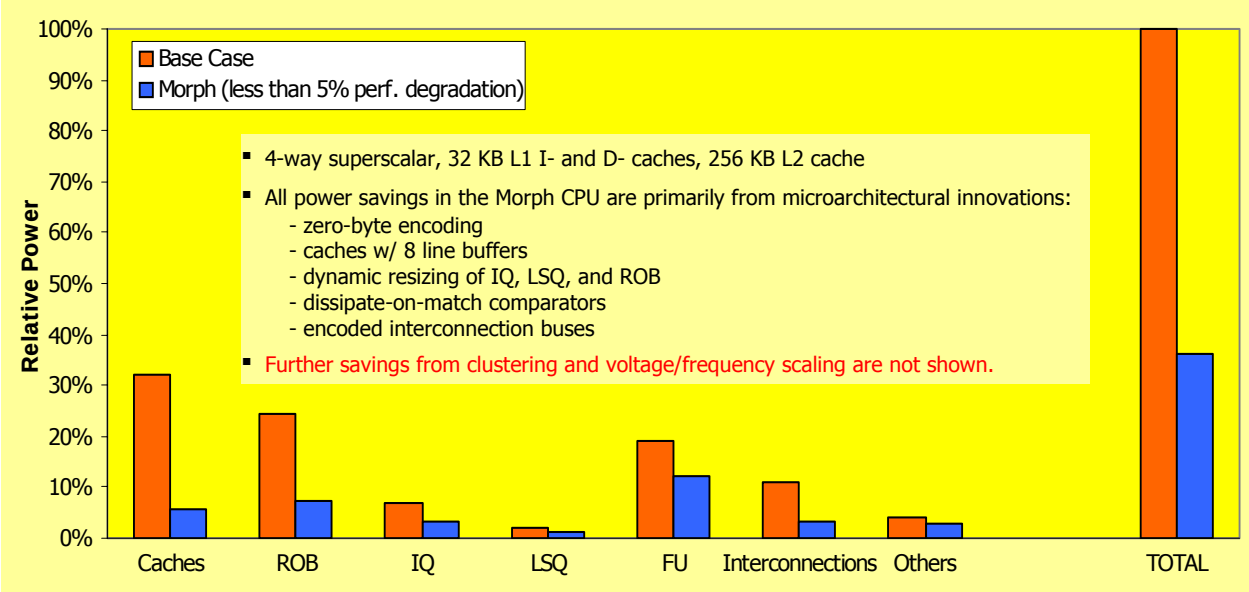


Figure 12. Relative power dissipations in the Morph CPU compared to the base case on a Component-by-component basis (clock power included within each major component)

7.2 Multi-Cluster

Evaluating an architecture such as RuDRA is difficult on an individual program, because you never know when, and for how long, different configurations ought to be employed. Consequently, the approach taken in Morph was to use the trace data from the testbed to simulate complete mission scenarios at a high level. There were three steps to this analysis process. First was processing of the time stamp logs from the test bench to the point where intervals of idle time could be identified. Second was development of sets of EPI/IPC configuration files that represented each of the different power management techniques. Finally, these two were combined to simulate a run-time that could dynamically change the CPU cluster configuration to optimally minimize energy usage.

The analysis process started with the Deep Impact scenario timestamp logs discussed earlier. These were first post-processed to replace all calls to the logging and simulation support routines by idle periods (since in a real flight such activities would not be present), and to collapse neighboring idle periods together into single longer idle periods. Thus each idle period is surrounded on both sides by the description of a task that was called. Each task entry was also augmented by both the task priority and overall execution time.

From these traces, sets of processing “intervals” were identified, where one interval starts at the end of an idle period, after which a string of one or more real tasks were executed, and terminated with some other idle period. For each such interval, the overall busy and idle periods are computed. The key statistic to take from each such interval is the ratio of the total interval time to the processing time. This ratio indicates by how much a factor the CPU’s performance could be slowed down during the processing period to eliminate the idle period.

The second step involved computing for each of three comparative power management techniques the potential power savings, along with effects on performance. The RuDRA specific techniques simulated included:

- variation of number of clusters only, with a constant clock and Vdd.
- variations of clusters and clock, with constant Vdd.
- variation of all three, with the Vdd to power changes mirroring what was observed from an Athalon microprocessor that included dynamic voltage and frequency control.

As a baseline, some conventional techniques using only Vdd, only clock, and then both, were also included. In all cases, both the power and energy savings were normalized onto a relative basis so that "peak performance" would be unity, and thus relative power savings could be judged. This process was most complex for the variable cluster configuration, where extensive simulations were necessary for both variations in active cluster count, and in relative access times to memory. Figure 13 graphs these configuration points for all six techniques. Note that the Full/Doze configuration only has two points: one at 100% performance (Full), and one at 0% (Doze).

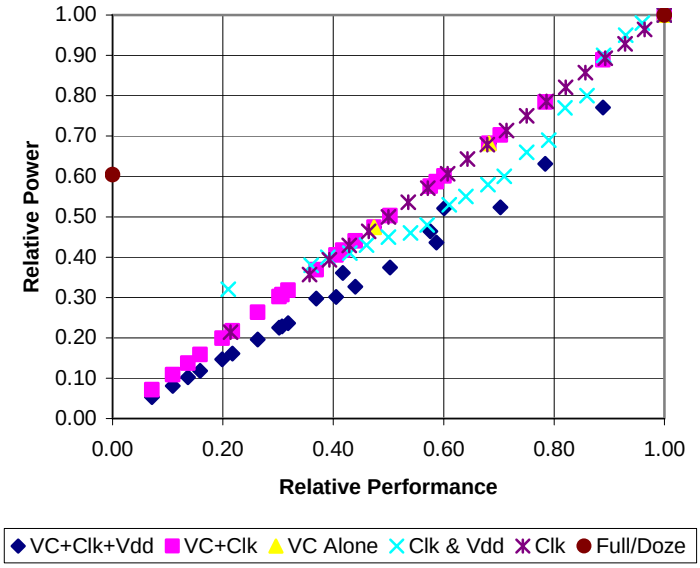


Figure 13. Power/Performance Configuration Points

It should be noted that in general the RuDRA variable cluster approaches are the best in terms of relative power per performance, and in fact are the only techniques that allow switching the machine into very low performance regions - exactly the regions where we expect to see real computers to be spending the bulk of their time on a real mission.

The final step was to build a simulator that would process the intervals one at a time, and for each interval determine for each management technique which configuration would provide the best energy savings, given the idle time available in the current

interval. Such savings were then integrated over all intervals to compute overall energy for the scenario.

While this corresponds to an "oracle" scheduler, where the end of each period is known (particularly the idle time available), it does give a solid upper bound on the maximum potential of each approach. Further, any real-world idle predictor would tend to affect all such management techniques equally, keeping the relative benefits roughly the same. Experimentation with some more realistic idle predictors has begun, with some relatively good correlation with the results presented here.

The key output of the exercise is a projection of the overall energy savings over the time period of each scenario. Figure 14 summarizes results for all scenarios and all six techniques. The "Energy Reduction Factor" is the factor by which the total energy expended over the scenario is reduced using that technique over a baseline full power, constant full performance run.

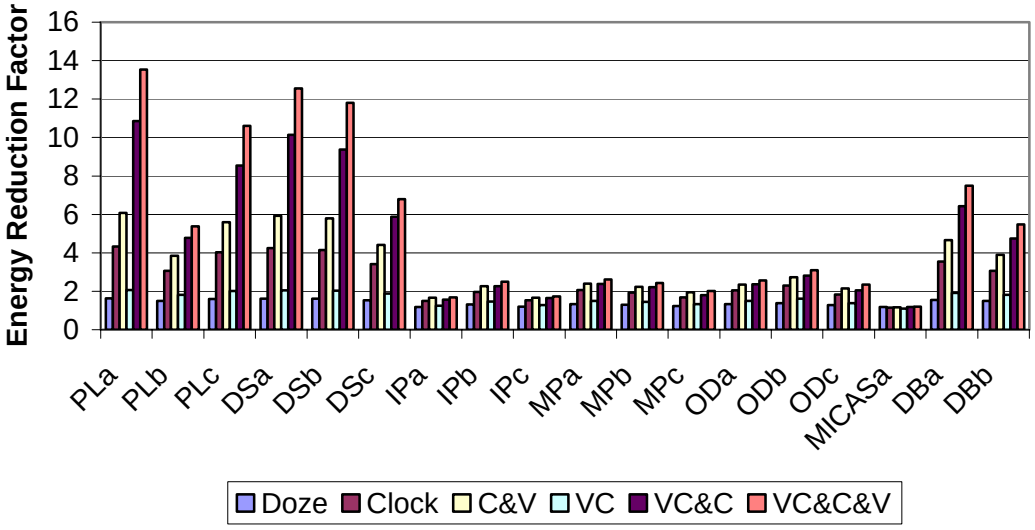


Figure 14. **Energy Reduction Factors as a Function of Scenario**

As can be seen, the variable cluster technique, when paired with clock reductions or clock and Vdd reductions, handily beats all other approaches for virtually all scenarios, and sometimes by large factors.

Perhaps even more revealing is Figure 15, where these energy reduction factors are graphed against the baseline processing load of each scenario. Here the advantages of the variable cluster techniques become even more pronounced, especially for those scenarios where the overall processor load is quite small, especially below 10%. This is exactly the region where we expect most of a real mission to be spent, indicating that over the long run, the technique can in fact deliver significant energy, and power, savings to the spacecraft.

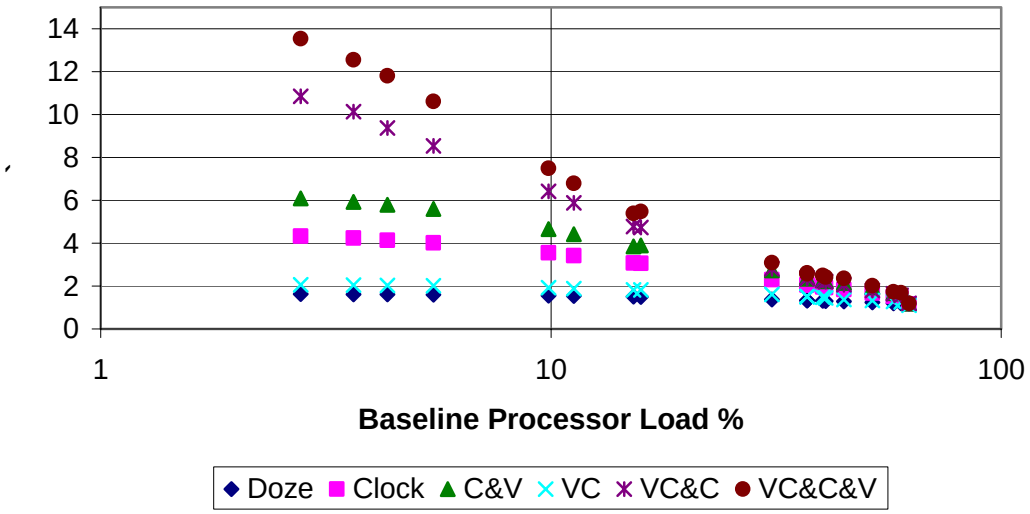


Figure 15. **Energy Reduction Factors as a Function of Processor Load**

In addition, the variable cluster techniques are superior, even during those scenarios of very high usage.

7.3 Putting It All Together

As a final exercise, we combined all these techniques into a very preliminary estimate of overall savings, pictured in the following table. Here we start with a baseline PPC chip with L2 cache, and memory as found in the testbed. To each of these we applied a series of estimated power reduction factor based on the above analyses. First were the independent factors such as dataflow improvements. Next, for the CPU we assumed a 90/10 ratio processing at low IPC versus high IPC, and developed power reduction numbers based on the RuDRA results. This gave about a 6X power reduction, without Vdd scaling. Factoring in Vdd scaling gave the potential for another factor of 4.

	CPU	L2	Mem	Watts
Baseline JPL PPC SBC (W)	2.9	2.4	2.3	7.6
Independent Factors	3	2.5	2	3.1
Var. Cluster (Hi IPC)	1.5	1	1	6.6
Var. Cluster (Lo IPC)	3	3	3	2.5
Combined Hi IPC	3	2.5	2	3.1
Combined Lo IPC	6	7.5	6	1.2
90/10 Rule-constant clock	5.7	7	5.6	1.3
Equivalent Power (W)	0.51	0.34	0.41	1.26
16X Clk - 2X Vdd	64	16	16	
90/10 w'clk scale (W)	0.103	0.114	0.137	0.354

Table 2. Preliminary Estimate of Overall Savings

One caveat is that this does not consider static power leakage, but the overall numbers are still compelling.

8.0 Specific Recommendations

Based on our study of the highly dynamic nature of the Deep Impact Applications on our testbed and our assessments of the various power savings techniques developed in the current effort, we offer the following recommendations to the designers of the next generation of computing systems for the deep space probes:

1) Use the following techniques in the order given to reduce the power requirements of the processor:

- Zero-byte encoding
- Caches with multiple line buffers and facilities for leakage reduction
- Dynamic allocation of datapath resources

- all of these techniques can be retrofitted into an existing design relatively easily.

2) Use voltage scaling and frequency scaling wherever possible in conjunction with a suitable kernel facility that takes into account real-time constraints.

3) Develop and use a data allocation module that places data intelligently in the memory hierarchy.

4) Expose the dynamic resource allocation hooks to the compiler and develop a compiler-based technique to match resource allocations to various statically-analyzed phases of the compiler.

5) Accommodate any additional processing need by using a variable cluster design and/or by increasing the sizes of the on-chip caches. The amount of hardware changes needed by this technique is relatively more complex.

As the next step along this path, we feel strongly that a three-level approach to developing some demonstration chips is most reasonable. These projected chips include:

- •An L2 cache macro designed not for density or speed, but to demonstrate the energy savings possible by the techniques developed in Morph. This macro would be packaged in a chip format that allows insertion into the L2 cache slot of JPL's PPC SBC test bench.
- •A modified baseline core with the dynamic zero-suppression and resource allocation techniques developed by Morph.
- •A variable cluster version of the above, built up from multiple copies of the modified baseline core, and coupled with the low power cache macros for the on-chip L1 caches.

9.0 Team Members

The project involved three organizations, the Computer Science and Engineering Department of the University of Notre Dame, the Computer Science Department of the State University of New York at Binghamton (a major campus of the State University of New York system), and the Center for Integrated Space Microsystems (CISM) of the Jet Propulsion Laboratory (JPL) of the California Institute of Technology. Notre Dame served as lead organization.

For Notre Dame, the PI was Dr. Peter M. Kogge. Other faculty members included Dr. Vincent Freeh (now at North Carolina State University) and Jay Brockman. Graduate students active on the project included Victor Zyuban, Arun Rodrigues, Virgil Andronache, Jason Zawodny, and Robert J. Minerick.

For SUNY Binghamton the PI was Dr. Kanad Ghose, with graduate students Dmitry V. Ponomarev, Gurhan Kucuk, Andrew Flinders, and Oguz Ergin.

For JPL the PI was Dr. Nikzad (Benny) Toomarian, who was assisted by Nazeeh Aranki, with significant participation by Jeffrey Nankung, Jagdish Patel, Sanjay Patel, Savio Chau, and Mohammed Mojarradi.

REFERENCES

[BTM 00] Brooks, D., Tiwari, V., Martonosi, M., "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", Proc. of the 27th Int'l Symposium on Computer Architecture, 2000.

[BA 97] Burger, D., and Austin, T. M., “The SimpleScalar tool set: Version 2.0”, Tech. Report, Dept. of CS, Univ. of Wisconsin-Madison, June 1997 and documentation for all Simplescalar releases (through version 3.0).

[BM 99] Brooks, D. and Martonosi, M., “Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance”, Proc. HPCA, 1999.

[Cai 99] Cai, G., "Architectural Level Power/Performance Optimization and Dynamic Power Estimation", Proc. of the CoolChips tutorial. An Industrial Perspective on Low Power Processor Design in conjunction with MICRO-32, 1999.

[GK 99] Ghose, K. and Kamble, M.B., “Reducing Power in Superscalar Processor Caches with Subbanking, Multiple Line Buffers and Bitline segmentation”, in Proc. ACM Int’l. Symposium on Low Power Electronic Design, 1999 (ISLPED 99), pp. 70-75.

[Gh 00] Ghose, K., “Reducing Energy Requirements for Instruction Issue and Dispatch in Superscalar Processors”, in Proc. ACM ISLPED 2000, pp. 231-234.

[GPK+ 00] Ghose, K., Ponomarev, D.P., Kucuk, G., Flinders, A., Kogge, P.M. and Toomarian, N., “Exploiting Bit-Slice Inactivities for Reducing Energy Requirements of Superscalar Processors”, in Proc. KOOLchips Workshop, held in conjunction with Micro-33, Dec., 2000.

[GPK+ 01] Kanad Ghose, Dmitry Ponomarev and Gurhan Kucuk, "Power Reduction in Superscalar Datapaths Through Dynamic Bit-Slice Activation" in Proc. *International Workshop "Innovative Architecture for Future Generation High-Performance Processors and Systems" (IWIA'01)*, 2001, pp.16-24.

[Itoh 96] Itoh, K., “Low Power Memory Design”, in Low Power Design Methodologies, ed. by Rabaey, J. and Pedram, M., Kluwer Academic Pub., pp. 201-251.

[KEP+ 03a] Gurhan Kucuk, Oguz Ergin, Dmitry Ponomarev, Kanad Ghose, "Distributed Reorder Buffer Schemes for Low Power", in Proc. *21st International Conference on Computer Design (ICCD'03)* , San Jose, October 2003, pp.364-370.

[KFG+ 00] Peter M. Kogge, Vincent W. Freeh, Kanad Ghose, Nikzad Toomarian, Nazeeh Aranki, “Morph: Adding an Energy Gear to a High Performance Microarchitecture for Embedded Applications”, in Proc. Kool Chips Workshop, MICRO-33, Monterey, CA, Dec. 10, 2000, pp. 9-16.

[KPG 02] Gurhan Kucuk, Dmitry Ponomarev, Kanad Ghose "Low-Complexity Reorder Buffer Architecture" , in Proc. *16th ACM International Conference on Supercomputing (ICS'02)* , New York, June 2002, pp. 57-66.

[KPE+ 03b] Gurhan Kucuk, Dmitry Ponomarev, Oguz Ergin, Kanad Ghose, "Reducing Reorder Buffer Complexity Through Selective Operand Caching", in Proc. *International Symposium on Low-Power Electronics and Design (ISLPED'03)*, Seoul, South Korea, August 2003, pp. 235-240.

[KNA+ 03a] Kogge, P.M., Namkung, J., Aranki, N., Toomarian, N. and Ghose, K., "A Comparative Analysis of Power and Energy Management Techniques in Real Embedded Applications,", Proc. IEEE IWIA'03, Kauai, HI, Jan. 2003.

[KNA+ 03b] Kogge, P.M., Namkung, J., Aranki, N., Toomarian, N. and Ghose, K., "Characterization of Future Deep Space Computing Loads", in Proceedings of Space Mission Challenges for Information Technology 2003 (SMC-IT 2003), Pasadena, CA, July 2003.

[KGPK 01] Kucuk, G., Ghose, K., Ponomarev, D., Kogge, P., "Energy Efficient Instruction Dispatch Buffer Design for Superscalar Processors", in Proc. of Int'l Symposium on Low-Power Electronics and Design, 2001, pp.237-242.

[KPE+ 04a] Gurhan Kucuk, Dmitry Ponomarev, Oguz Ergin, Kanad Ghose, "Complexity-Effective Reorder Buffer Designs for Superscalar Processors", to appear in the IEEE Transactions on Computers (June 2004).

[MFK 02] Robert J. Minerick, Vincent W. Freeh, and Peter M. Kogge. Dynamic power management using feedback. In Proceedings of *Workshop on Compilers and Operating Systems for Low Power*, pp 6-1--6-10, Charlottesville, Va, September, 2002.

[M 02] Robert J. Minerick, "Feedback-Directed, Adaptive Energy Control," M.S. Thesis, CSE Dept., Univ. of Notre Dame, Nov. 2002

[PKE+ 03a] Dmitry Ponomarev, Gurhan Kucuk, Oguz Ergin, Kanad Ghose "Power Efficient Comparators for Long Arguments in Superscalar Processors", in Proc. *International Symposium on Low-Power Electronics and Design (ISLPED'03)* , Seoul, South Korea, August 2003, pp. 378-383.

[PKE+ 03b] Dmitry Ponomarev, Gurhan Kucuk, Oguz Ergin, Kanad Ghose, "Reducing Datapath Energy Through the Isolation of Short-Lived Operands", in Proc. *12th International Conference on Parallel Architectures and Compilation Techniques (PACT'03)* , New Orleans, September 2003, pp.258-268.

[PKE+ 03c] Dmitry Ponomarev, Gurhan Kucuk, Oguz Ergin, Kanad Ghose, Peter Kogge "Energy-Efficient Issue Queue Design", in IEEE Transactions on VLSI Systems, Vol 11, No. 5 (October 2003). pp. 789-800.

[PKE+ 04a] Dmitry Ponomarev, Gurhan Kucuk, Oguz Ergin, Kanad Ghose, "Energy-Efficient Comparators for Superscalar Datapaths", to appear in the IEEE Transactions on Computers (May 2004).

[PKE+ 04b] Dmitry Ponomarev, Gurhan Kucuk, Oguz Ergin, Kanad Ghose, "Isolating Short-Lived Operands for Energy Reduction", to appear in the IEEE Transactions on Computers

[PKG 01] Ponomarev, D. P., Kucuk, G., and Ghose, K., “Reducing Power Requirements of Instruction Scheduling Through Dynamic Allocation of Multiple Datapath Resources”, in Proc. of the 34th IEEE/ACM International Symposium on Microarchitecture (MICRO-34), Austin, TX, December 2001, pp. 90-101.

[PKG 01a] Ponomarev, D., Kucuk, G., Ghose, K., “Power Reduction in Superscalar Datapaths Through Dynamic Bit-Slice Activation”, in Proc. of Int'l. Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, (IWIA'01), 2001, pp.16-24.

[PKG 01b] Ponomarev, D., Kucuk, G., Ghose, K., “Dynamic Allocation of Datapath Resources for Low Power”, in Proc. of Workshop on Complexity-Effective Design, held in conjunction with ISCA-28, June, 2001.

[PKG 02] Ponomarev, D. P., Kucuk, G., and Ghose, K., “Accupower: An accurate power estimation toll for superscalar processors”, in Proc. of the DATE 2002 Symposium, pp. 124-129.

[R 03] Arun Rodrigues, "RuDRA: A Reactive Dissipation Reducing Architecture," M.S. Thesis, CSE Dept., Univ. of Notre Dame, April 28, 2003

[VZA 00] Villa, L., Zhang, M. and Asanovic, K., “Dynamic Zero Compression for Cache Energy Reduction", in Proc. Micro-33, Dec. 2000.

[Z 01] Zawodny, Jason and Peter M. Kogge, "Cache In Memory," International Workshop on Innovative Architecture 2001 (IWIA01), Maui High Performance Computer Center, Maui, HI, Jan. 18-19, 2001.

[ZK 00] Zyuban, V. and Kogge, P., "Optimization of High-Performance Superscalar Architectures for Energy Efficiency", in Proc. of Int'l Symposium on Low-Power Electronics and Design, 2000, pp. 84-89.

[ZK 01] Zyuban, V. and Kogge, P.M., “Inherently Lower-Power High-Performance Superscalar Architectures,” IEEE Trans. On Computers, March 2001.